Preconditioning for Feature Selection in Classification

by

Jani Pretorius

Thesis presented in partial fulfilment of the requirements for the degree of Master of Commerce in Mathematical Statistics in the Faculty of Economic and Management Sciences at Stellenbosch University

Supervisor: Prof. S.J. Steel

April 2019

Stellenbosch University https://scholar.sun.ac.za

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

Copyright © 2019 Stellenbosch University
All rights reserved.

Stellenbosch University https://scholar.sun.ac.za

Abstract

Preconditioning for Feature Selection in Classification

J. Pretorius

Thesis: MComm (Mathematical Statistics)

April 2019

Increased dimensionality of data is a clear trend that has been observed over the past few decades. However, analysing high-dimensional data in order to predict an outcome can be problematic. In certain cases, such as when analysing genomic data, a predictive model that is both interpretable and accurate is required. Many techniques focus on solving these two components simultaneously; however, when the data are high-dimensional and noisy, such an approach may perform poorly. Preconditioning is a two-stage technique that aims to reduce the noise inherent in the training data before making final predictions. In doing so, it addresses the issues of interpretability and accuracy separately. The literature on this technique focuses on the regression case, but in this thesis, the technique is applied in a classification setting.

An overview of the theory surrounding this method is provided, as well as an empirical analysis of the method. A simulation study evaluates the performance of the technique under various scenarios and compare the results to those obtained by standard (non-preconditioned) models. Thereafter, the models are applied to real-world datasets and their performances compared.

Based on the results of the empirical work, it appears that, at their best, preconditioned classifiers can only reach a performance that is on par with standard classifiers. This is in contrast to the regression case, where the literature has shown that preconditioning can outperform standard regression models in high-dimensional settings.

Stellenbosch University https://scholar.sun.ac.za

Uittreksel

Prekondisionering vir Veranderlike Seleksie in Klassifikasie

J. Pretorius

Tesis: MComm (Wiskundige Statistiek)

April 2019

'n Toename in die dimensionaliteit van datasetelle is 'n duidelike tendens wat oor die afgelope paar dekades na voorskyn gekom het. Om hoër-dimensionele data te analiseer sodat 'n uitkoms voorspel kan word, kan problematies wees. In sekere gevalle, soos wanneer genetiese data geanaliseer word, word 'n voorspellende model wat beide interpreteerbaar, sowel as akkuraat is, verlang. Baie tegnieke fokus daarop om hierdie twee aspekte gelyktydig op te los, maar wanneer die data van 'n hoë dimensie is en geruis bevat, kan hierdie benadering swak resultate oplewer. Prekondisionering is 'n twee-fase prosess wat daarop gemik is om die geruis in die afrigdatastel te verminder voordat 'n finale voorspelling gemaak word. Sodoende spreek dit die kwessies van interpreteerbaarheid en akkuraatheid afsonderlik aan. In die literatuur word daar klem gelê op die regressie geval. In hierdie tesis word die tegniek egter toegepas in 'n klassifikasie konteks.

'n Oorsig van die teorie aangaande hierdie metode word verskaf, sowel as empiriese studies. Simulasie studies evalueer die prestasie van die tegniek onder verskeie omstandighede en vergelyk die uitkomste met dié wat deur standaard (nie-geprekondisioneerde) modelle behaal was. Daarna word die modelle toegepas op regte-wêreld datastelle en hul resultate vergelyk.

Gebaseer op die resultate van die empiriese werk wil dit blyk asof geprekondisioneerde klassifikasiemodelle, op hul beste, slegs so goed as standaard klassifikasiemodelle kan presteer. Hierdie bevindinge staan in kontras met die regressie geval, waar die literatuur wys dat prekondisionering standaard regressiemodelle kan uitpresteer in hoë dimensionele gevalle.

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

I would like to express my sincerest gratitude to Prof. S.J. Steel for his mentorship over the past 2 years, for always being patient and understanding and for his attention to detail in the preparation of this document.

To my husband, Arnu Pretorius, I would like to express my immense appreciation and admiration for the support he has given me, both emotionally and academically.

Finally, I would like to thank my family for their encouragement and for their unwavering confidence in me.

Stellenbosch University https://scholar.sun.ac.za

Contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of Figures

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of Tables

Stellenbosch University https://scholar.sun.ac.za

Frequently Used Notation

Inputs (predictors)

X: Random scalar input variable or the design matrix. Its meaning will be clear from the context.

x: Observed value of the random input variable X.

$\boldsymbol{X}$: Random $p \times 1$ vector of inputs, *i.e.* $\boldsymbol{X}^T = [X_1, ..., X_p]$.

$\boldsymbol{x}_i$: The i-th observed $p \times 1$ vector of inputs, *i.e.* $\boldsymbol{x}_i^T = [x_{i1}, ..., x_{ip}]$, *i.e.* an observed data point in p-dimensional space. Sometimes this vector is prepended with a one, which represents the intercept term, so that $\boldsymbol{x}_i^T = [1, x_{i1}, ..., x_{ip}]$. It follows that $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^T$.

$\boldsymbol{x}_j$: The $1 \times N$ vector consisting of the observed values of the j-th input variable for all observations, *i.e.* $\boldsymbol{x}_j^T = [x_{1j}, ..., x_{Nj}]$. It follows that the design matrix can also be expressed as $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p]$.

Outputs (responses)

Y: Random scalar response variable.

y: Observed value of the output variable Y.

$\hat{y}$: Estimated value of the output variable y.

$\mathbf{Y}$: Vector of random response variables.

k/K: Number of classes for a categorical response.

Data

N: The number of observations in a dataset.

p: The total number of predictors/input variables in a dataset.

r: The number of predictors/input variables in a dataset that carry signal, i.e. of those that have a direct relationship with the response. $r \leq p$.

ϵ/ε: A random error term which is independent of the predictors and has an expected value of zero. Referred to interchangeably as noise throughout this dissertation.

$\mathcal{T}$: Random set of training data represented as a set $\{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$.

$\mathcal{P}$: The set of predictors that have a relationship with the outcome.

Stellenbosch University https://scholar.sun.ac.za

$\hat{\mathcal{P}}$: The estimate of set $\mathcal{P}$.

$p_{\hat{\mathcal{P}}}$: The number of predictors in the set $\hat{\mathcal{P}}$.

Stellenbosch University https://scholar.sun.ac.za

Functions

$f(\cdot)$: A function mapping inputs to outputs, capturing the relationship between these two.

$I(\cdot)$: The indicator function which is equal to 1 when the argument is true and equal to 0 otherwise.

$\hat{f}(\cdot)$: An estimate of the function $f(\cdot)$.

$C(\cdot)$: A classification rule mapping $C(\cdot) \rightarrow \{1, 2, \cdots, K\}$.

s_j: A score statistic measuring the strength of the relationship between X_j and Y.

$\mathcal{G}$: The proportion of signal-carrying predictors selected, calculated as $\frac{1}{p_{\hat{\mathcal{P}}}} \sum_{j=1}^{p} I(X_j \in (\mathcal{P} \cap \hat{\mathcal{P}}))$

Probability

$P(X)$: The probability distribution of the random variable X.

$P(X, Y)$: The joint distribution of X and Y.

$P(X = x)$, or $P(x)$: The probability that the random variable X takes on the value x.

$P(Y = k|\boldsymbol{X} = \boldsymbol{x})$, or $P(k|\boldsymbol{x})$: The conditional probability that the random variable Y takes on the qualitative value k given that the random vector $\boldsymbol{X}$ has taken on the realisation $\boldsymbol{x}$.

$\hat{P}(Y = k|\boldsymbol{X} = \boldsymbol{x})$, or $\hat{P}(k|\boldsymbol{x})$: The estimated conditional (posterior) probability that the random variable Y takes on the qualitative value k given that the random vector $\boldsymbol{X}$ has taken on the realisation $\boldsymbol{x}$, as estimated by a classification algorithm trained on $\mathcal{T}$.

Stellenbosch University https://scholar.sun.ac.za

List of Abbreviations and Acronyms

ISPC Iterative Supervised Principal Components
KNN K-Nearest Neighbour
LAR Least Angle Regression
LASSO Least Absolute Shrinkage and Selection Operator
LDA Linear Discriminant Analysis
LR/LogReg Logistic Regression
MSE Mean Squared Error
MLE Maximum Likelihood Estimators
NSC Nearest Shrunken Centroids
PC Principal Component
PCA Principal Component Analysis
PCR Principal Component Regression
PLR Penalised Logistic Regression
PLS Partial Least Squares
SIS Sure Independence Screening
SNR Signal-to-Noise Ratio
SPCA Supervised Principal Component Analysis
SRBCT Small Round Blue Cell Tumour
SVD Singular Value Decomposition

Stellenbosch University https://scholar.sun.ac.za

Chapter 1

Introduction

Data are being generated at a faster rate than ever before and data sets are becoming larger as new, faster and better technology become available in a multitude of different devices, all collecting and generating data. In isolation, data has little value, but when combined with analytics, the information gained can be invaluable. With such quantity of data however, there is a need for data analysis to be semi-automated.

Statistical Learning is a field of statistics that focuses on learning from data. One stream within statistical learning is supervised learning, which involves developing models to make accurate predictions by learning from input-output examples. Applications for supervised learning are numerous, ranging from estimating future dam-water levels based on the current hydro-climatic conditions (Obringer and Nateghi, 2018), to determining whether a brain tumour is present on a given MRI scan (Gamage, 2017). With the recent improvements in technology, many of the previously computationally cumbersome techniques have become less 'expensive' and as a consequence, have become more common-place.

Another stream is unsupervised learning. Here only input data are provided and the goal is to detect some hidden structure within the data to gain a better understanding of the true underlying data-generating mechanism. An example of this is to determine how similar or different the online-shopping motivations of people in different age groups, income brackets or of different genders are and which of these variables causes the greatest separation between groups (Ganesh *et al.*, 2010). Unsupervised learning is not as well-developed as supervised learning, since there is no clear target to measure the performance of the model against.

When the number of predictor variables in a dataset is much larger than the number of observations, several problems are encountered - see for example Chapter 18 of Hastie *et al.* (2009), for a discussion. These are discussed in the

Stellenbosch University https://scholar.sun.ac.za

sections to come, as well as techniques that are used to work around the problems of high-dimensional data. This thesis focuses specifically on a supervised learning technique called preconditioning, and its application to a binary classification problem. The technique relies on concepts which initially stemmed from an unsupervised learning context and these will also be elaborated on below.

In the sections that follow, some of the terminology and mathematical concepts relating to statistical learning are introduced. The motivation for the research and thesis outline are also provided.

1.1 Concepts, Terminology and Notation

1.1.1 Supervised Learning

Supervised learning involves approximating the relationship between independent variables and a dependent variable by the use of a statistical model. The model is based on a training set, $\mathcal{T}$, consisting of N observations of input-output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$. Let the design matrix $X : N \times p = [\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_p]$ denote the input dataset with N observations and p variables. Each of the p vectors of input variables can be expressed as $\mathbf{x}_j : N \times 1 = [x_{1j}, x_{2j}, ..., x_{Nj}]^T$. The dependent variable observations can be summarised in a vector $\mathbf{y} : N \times 1 = [y_1, y_2, ..., y_N]^T$. Please note that when these symbols are used in lower-case, they refer to an observed set of values, whereas the upper-case equivalents refer to the more generic random variables.

The broad goal of supervised learning is to estimate the conditional probability distribution $P(y|\mathbf{x})$. One way of doing so is by the generative approach. Here the approach is to create a model of the joint probability distribution first, and then to condition on $\mathbf{x}$ to ultimately derive $P(y|\mathbf{x})$. Another approach is to model the conditional probabilities directly. This is referred to as the discriminative approach. The latter is the approach that is taken in the sections below.

The values of the response variable can be numerical or categorical. For a numerical response variable, the outcome variable will be discrete or continuous, i.e., $y \in \mathbb{R}$ and therefore regression techniques are used to obtain a model. A linear regression model is a simple form of such a model, but is also one of the most widely used models (Murphy, 2012, p. 19). The idea is that one can approximate the response by assuming that it is linearly related to the input variables as follows:

$$Y = \beta_0 + \beta_1^T \mathbf{x} + \epsilon$$

where ϵ is the random variation that is not explained by the predictors. The expected value of these error terms should be zero to ensure that all the sys-

Stellenbosch University https://scholar.sun.ac.za

temic information is captured in the intercept term and/or by the relationship between the input variables and the response. It is often assumed that $\epsilon \sim N(0, \sigma^2)$, where the noise is sampled independently and identically. It then follows that $y_i|\mathbf{x}_i$ are observations drawn from $Y_i|\mathbf{X}_i = \mathbf{x}_i \sim N(\mu(\mathbf{x}_i, \beta), \sigma^2)$, for each $i = 1, 2, \cdots, N$ and where $\mu(\cdot)$ is a parametrised function of $E(Y|\mathbf{x})$, with coefficients, β. Under the above assumptions, the expected value, $\mu(\mathbf{x}, \beta)$, can be written as:

$$E(Y|\mathbf{x}) = \mu(\mathbf{x}, \beta) = \beta_0 + \beta_1^T \mathbf{x},$$

which in vectorised form is given by

$$E(Y|\mathbf{x}) = \beta^T \mathbf{x}^*,$$

where $\mathbf{x}^* = [1\ \mathbf{x}]$ and $\beta = [\beta_0\ \beta_1^T]^T$. The task now is to find the parameters β which maximise the likelihood that the response vector $\{y_1, y_2, \cdots, y_N\}$ was observed, conditional on the given input variables $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. Assuming that the variance of the distribution is constant (w.r.t. the input space), the likelihood function is given by:

$$\begin{aligned}
L\big(\theta = \{\beta, \sigma\}\big) &= \prod_{i=1}^{N} P(y_i|\mathbf{x}_i, \theta) \\
&= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - \mu(\mathbf{x}_i, \beta))^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N e^{-\sum_{i=1}^{N} \frac{(y_i - \mu(\mathbf{x}_i, \beta))^2}{2\sigma^2}} \\
log\left(L(\theta)\right) &= -\sum_{i=1}^{N} \frac{(y_i - \mu(\mathbf{x}_i, \beta))^2}{2\sigma^2} + \text{const.} \\
&\propto -\sum_{i=1}^{N} (y_i - \mu(\mathbf{x}_i, \beta))^2 \\
&\propto -\frac{1}{N} \sum_{i=1}^{N} (y_i - \mu(\mathbf{x}_i, \beta))^2.
\end{aligned}$$

In the fourth line of the equation, the constant terms were dropped since they do not impact the Since maximising the log-likelihood is equivalent to minimising the negative log-likelihood and the estimate of $E(Y|\mathbf{x})$ is $\hat{y}_i = \mu(\mathbf{x}_i, \hat{\beta})$, it follows that the objective function to optimise, i.e., $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$, is the mean of squared error (MSE). This is commonly referred to as squared error loss.

Stellenbosch University https://scholar.sun.ac.za

For a linear model, a closed-form solution for the maximum likelihood estimators (MLE) can be obtained. The negative log-likelihood can be rewritten as follows:

$$\frac{1}{2}(\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta) = \frac{1}{2}\beta^T X^T X\beta - \beta^T X^T \mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y},$$

where the first column of X is $\mathbf{1}$. Taking the derivative with respect to β and setting the resulting expression equal to zero, we obtain:

$$X^T X\hat{\beta} - X^T\mathbf{y} = 0 \tag{1.1}$$

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}, \tag{1.2}$$

provided that $X^T X$ is non-singular so that the inverse exists. This estimate can now be used to compute a predicted response for a previously unseen case, say $\mathbf{x}_0$, as follows:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1{}^T \mathbf{x}_0.$$

Whether or not the model predictions are accurate will depend on the validity of the assumptions made regarding the distribution of the response variable/noise term, linearity, as well as the representativeness of the training sample.

Depending on the assumed conditional distribution of $Y_i|\mathbf{X}_i$, different loss functions can be obtained. For a categorical output, the outcome variable will be an indicator of class membership. In a multiclass problem where there are k classes, these can be represented by $Y_i \in \{1, 2, \cdots, k\}$, but for a two-class (or binary) problem the encoding $Y_i \in \{0, 1\}$ is commonly used. In categorical cases, classification techniques can be used. A common approach is to model the posterior probability of Y_i taking on a values associated with each of the given classes, by some model and then classify the unseen observation to the class with the highest posterior probability.

An example of such a model for a binary classification problem is logistic regression. Here one assumes that the probability that Y_i belongs to class 1 is linearly related to $\mathbf{X_i}$ through a link function. The link function ensures that the model output is restricted to the range $[0, 1]$, so that it resembles probability estimates. The logit transformation $\phi(z) = \frac{1}{1+e^{-z}}$ is a smooth function which maps $\mathbb{R} \to [0, 1]$ and can therefore be used as a link function in this case. This function is illustrated in Figure 1.1.

For simplicity of notation, let $P(Y_i = 1|\mathbf{x}_i, \beta)$ be denoted by π. The logistic

Stellenbosch University https://scholar.sun.ac.za

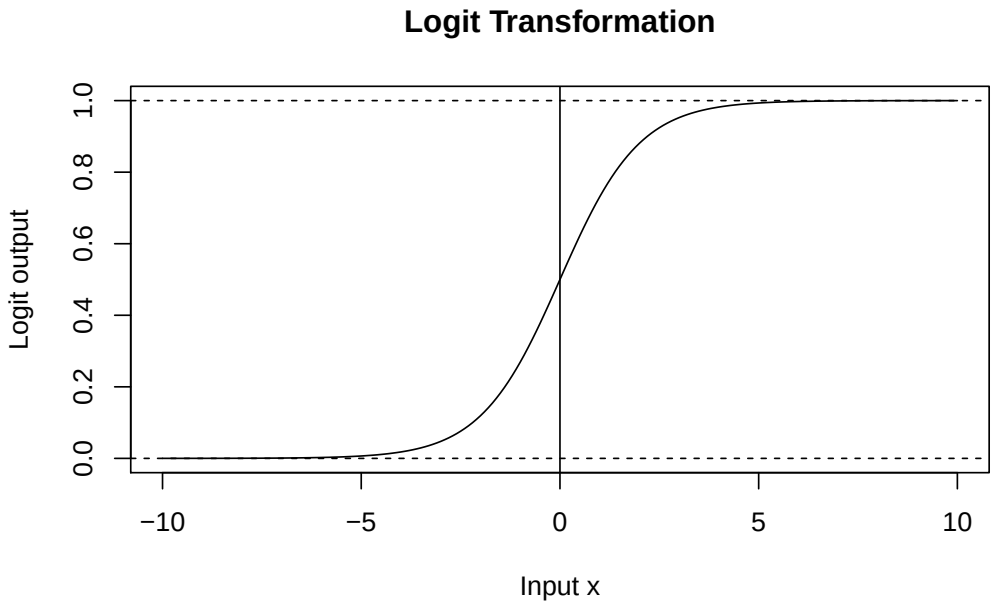

Figure 1.1: The logit function maps any real number in $\mathbb{R}$ to the range [0,1].

regression model can be derived as follows:

$$\begin{aligned}
\pi &= \phi(\beta_0 + \beta^T \mathbf{x}_i) \\
&= \frac{1}{1 + e^{-(\beta_0 + \beta^T \mathbf{x}_i)}} \\
\frac{1}{\pi} &= 1 + e^{-(\beta_0 + \beta^T \mathbf{x}_i)} \\
\frac{1-\pi}{\pi} &= e^{-(\beta_0 + \beta^T \mathbf{x}_i)} \\
-\log\left(\frac{1-\pi}{\pi}\right) &= \beta_0 + \beta^T \mathbf{x}_i \\
\log\left(\frac{\pi}{1-\pi}\right) &= \beta_0 + \beta^T \mathbf{x}_i.
\end{aligned}$$

To estimate the model parameters β, the likelihood function is considered. For a binary classification problem, one can assume that $Y_i|\mathbf{X}_i = \mathbf{x}_i \sim \text{Bernoulli}(p(\mathbf{x}_i, \beta))$, where $p(\cdot)$ is a function of $\mathbf{x}_i$ and a set of model parameters β and

Stellenbosch University https://scholar.sun.ac.za

$p(\mathbf{x}_i, \beta) = P(\mathbf{Y}_i = 1|\mathbf{x}_i, \beta)$. It follows that the likelihood function is:

$$\begin{aligned} L\big(\theta = \{\beta\}\big) &= \prod_{i=1}^{N} P(y_i|\mathbf{x}_i, \theta) \\ &= \prod_{i=1}^{N} p(\mathbf{x}_i, \beta)^{y_i}(1 - p(\mathbf{x}_i, \beta))^{1-y_i} \\ l(\theta) = log\left(L(\theta)\right) &= \sum_{i=1}^{N} [y_i log(p(\mathbf{x}_i, \beta)) + (1 - y_i) log(1 - p(\mathbf{x}_i, \beta))]. \end{aligned}$$

The log-likelihood function can be rewritten as a negative log-likelihood so that the maximisation problem is turned into a minimisation one. This loss function is also called cross-entropy. Unlike in the case of squared error loss, taking the derivative of the cross-entropy loss and setting it equal to zero will not yield an equation which permits a closed-form solution. The point at which the negative log-likelihood reaches a global minimum can instead be estimated by the use of first- or second-order optimisation techniques. The name refers to the order of derivative that is used in the iterative algorithm. Gradient descent is a first-order optimisation algorithm that starts at an initial point and takes a step in the direction of the negative gradient until convergence. Suppose ρ is the step size, then the updating rule is:

$$\beta^{(new)} = \beta - \rho \times \frac{\partial l(\beta)}{\partial \beta}.$$

The art lies in finding a suitable step size. Using a step size that is too small may take very long to converge, but using a step size that is too large may never lead to convergence.

Second-order optimisation solves this problem by making use of the second-order derivative information to determine the step size. If the curvature of the loss surface is large then presumably the algorithm has found a valley in the loss surface, so a smaller step should be taken so as to not overshoot the (local) minimum point. When the curvature is very subtle, then the parameter estimate is not near a valley in the loss function, so a larger step can be taken to speed up the convergence. There is a restriction that applies when working with second-order optimisation algorithms: the loss function must be strictly convex. This is to ensure that the step taken is always in the descent direction. An example of a second-order optimiser is the Newtown-Raphson algorithm, which iteratively computes the following update rule until convergence:

$$\beta^{(new)} = \beta - \left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}\right)^{-1} \frac{\partial l(\beta)}{\partial \beta}$$

.

Even where a closed form solution is available, as in linear regression, it is

Stellenbosch University https://scholar.sun.ac.za

sometimes still preferable to use an optimisation technique. One reason for this is that computing the inverse of the matrix $X^T X$ can be computationally intensive when p is large. Another reason is related to overfitting, which will be discussed in the next section.

Once the algorithm has converged, the parameter estimates can be substituted into the model equation in order to predict a response value for an unseen $\mathbf{x}_0$, i.e.,

$$\hat{p} = \frac{1}{1 + e^{-(\hat{\beta}_0 + \hat{\beta}^T \mathbf{x_0})}}.$$

In some cases only the predicted probability is required. At other times, a prediction regarding the final class of the response variable is needed. In this case one will assign a class label according to the label with the highest posterior probability, i.e., $\text{argmax}_k P(Y = k|\mathbf{X} = \mathbf{x})$.

Logistic regression can also be applied in a multi-class classification context and expressions similar to the above can be derived. In this case the assumed distribution of $Y_i|\mathbf{X}_i$ is a multinomial distribution where $N = 1$ (also recently referred to as a multinoulli distribution).

Returning to the regression case, a general model for expressing the relationship between the input variables and the output variable is formulated below:

$$\phi(Y) = f(\mathbf{X}) + \varepsilon \tag{1.3}$$

where $\phi(\cdot)$ represents any link function including the identity function. The unknown function $f(\cdot)$ represents the true underlying relationship between $\mathbf{X}$ and Y and ε represents the random error term which is independent of $\mathbf{X}$ and has $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. These two terms are often referred to as the signal and the noise respectively.

The challenge is to find a model which describes the signal accurately without modelling any of the noise. When limited data are available, however, distinguishing between the two is not an easy task. If model parameters are estimated by minimising a loss function based on the training sample, the minimum error will occur when a model fits the data perfectly. The problem is that this model might not perform well when evaluated on unseen data. This discrepancy between the error obtained on the training dataset and the error obtained on previously unseen data, is referred to as overfitting. Parametric approaches, such as the models above, are less susceptible to overfitting due to the structural assumptions made, whereas non-parametric approaches are particularly susceptible to overfitting. The issue of overfitting can be effectively illustrated by a simple non-parametric classifier called k-nearest neighbours

Stellenbosch University https://scholar.sun.ac.za

(KNN).

Let a target point be denoted by $\mathbf{x}_0$ and the set of k points in the training set that lie closest to $\mathbf{x}_0$ in distance be denoted by N_0^k. The set N_0^k is referred to as the target point's *k-nearest neighbours*. A KNN model classifies $\mathbf{x}_0$ to the majority class of the set N_0^k. The conditional probability estimate of belonging to class g can be expressed as

$$P(Y = g|\mathbf{X} = \mathbf{x_0}) = \frac{1}{k} \sum_{i \in N_0^k} I(y_i = g).$$

Figure 1.2 illustrates KNN decision boundaries for a binary classification problem when the number of neighbours used is varied. The data points for each class are generated from a combination of 10 different bivariate normal distributions a common covariance matrix of $\Sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 0,2 \end{bmatrix}$. The means of the 10 distributions are themselves generated from bivariate normal distributions. The means of the blue class are generated from $N\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}; \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$, while the means of the orange class are generated from $N\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}; \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$. The shaded regions indicate the class to which a hypothetical observation will be classified for each pair of values from X_1 and X_2. The two classes are indicated in blue and orange.

The misclassification rate, which is defined as

$$\frac{1}{N} \sum_{i=1}^{N} I(y_i \neq \hat{y}_i),$$

is an intuitive measure that can be used to evaluate the performance of a classifier. If computed on the training set, then of the three models illustrated, $k = 1$ will seem like the best model since it achieves a zero error rate on the training set. Increasing k leads to an increase in the training error rate, but provides a smoother decision boundary. The important caveat to note is that an improvement in performance on the training set does not necessarily extend to an increase in performance on unseen data. When this happens, overfitting has taken place. This phenomenon, along with underfitting, is illustrated in Figure 1.3. The quantity that is truly of interest is therefore the performance of the model on future, unseen data. This measure is called the generalisation error or the test error and it is defined as the expected value of the error rate, over all possible future data. Ideally one would like to minimise this quantity instead, but since full population data are not available, the generalisation error is not directly measurable. There are however ways in which it can be

Stellenbosch University https://scholar.sun.ac.za

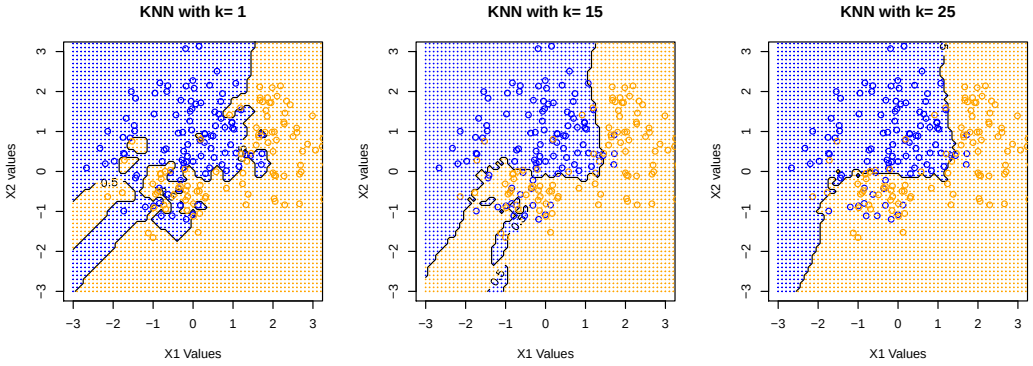

Figure 1.2: *k-nearest neighbours decision boundary on binary data.* **Left:** k=1 has zero misclassification rate on the training sample, since it fits the data perfectly. **Middle:** k=15 has a smoother decision boundary, but small isolated pockets classifying to the blue class still persist. **Right:** k=25 has the largest misclassification rate of the three, but seems to be less influenced by possible noisy data points. See also Chapter 2 of James *et al.* (2013).

approximated, one of which is described next.

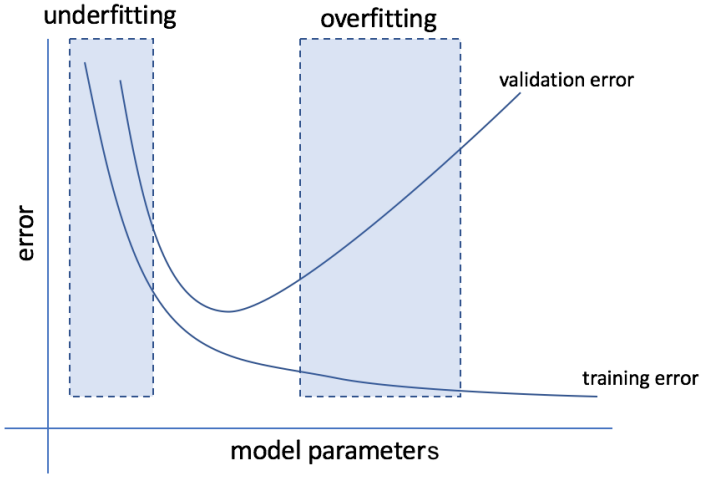

Figure 1.3: *Overfitting.* When test error increases as training error decreases, overfitting has taken place. Source: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/

If the training set $\mathcal{T}$ is fixed and the loss function is denoted by $L(Y, \hat{f}(\mathbf{X}))$,

Stellenbosch University https://scholar.sun.ac.za

where $\hat{f}(\cdot)$ is trained on $\mathcal{T}$, then the test error can be written as follows,

$$\text{Err}_{\mathcal{T}} = E[L(Y, \hat{f}(\mathbf{X}))|\mathcal{T}]. \tag{1.4}$$

Since $\mathcal{T}$ is fixed, $\text{Err}_{\mathcal{T}}$ refers to the error for a specific training set. To obtain an estimate of the error over any training set, the expectation can be taken with respect to all training datasets such that the expected test error can be expressed as follows,

$$\begin{aligned}\text{Err} &= E(\text{Err}_{\mathcal{T}}) \\ &= E_{\mathcal{T}}[E_{(X,Y)}[L(Y, \hat{f}(\mathbf{X}))|\mathcal{T}]] \\ &= E_{(X,Y,\mathcal{T})}[L(Y, \hat{f}(\mathbf{X}))].\end{aligned}$$

Hastie *et al.* (2009, p. 220) state that although the aim is to estimate $\text{Err}_{\mathcal{T}}$, it does not seem possible to estimate this quantity based on the same $\mathcal{T}$, whereas Err can be estimated effectively.

When enough data are available, a good approach to estimate this quantity is to divide the sample data randomly into three parts: the training set, the validation set and the test set. The samples should be selected so that their compositions are similar to that of the original sample. This can be done by ensuring that each observation has an equal probability of being selected. The training set is used to fit the model and find the model parameter estimates. The model complexity must then selected by minimising the error obtained when evaluating the model on the validation set. This will involve fitting multiple models of varying complexity, evaluating their performance on the validation set and selecting the one with the lowest error. Once the model complexity is selected, the final model is fit by combining the training and validation sets. The test set is only used once: to obtain an estimate of the accuracy of the final model. The error made on the test set will be the estimate of generalisation error. The proportion of data to use for each each step is not prescribed exactly and will vary depending on the size of the sample available; however it is common to use the majority of the data in the training step. When the sample is small, the model relies more heavily on each training observation and therefore overfitting is more likely. In this case a larger proportion of the data will be used in the training dataset, with the remainder being split between validation and test samples.

In cases where data are insufficient for this approach, the validation step can be approximated by efficient re-use of the training sample. Cross-validation is such a technique which directly estimates Err. This is done by averaging the approximate validation error over results based on varying training and validation sets. The sets are obtained by dividing the sample data randomly into k equally sized parts. At each iteration, one of the k subsets is omitted

Stellenbosch University https://scholar.sun.ac.za

from the training set and used, in lieu of the validation data, to assess the model. The average error rate over all the iterations is calculated and the final model is selected as the one yielding the minimum validation error. The cross-validation estimate of the prediction error curve, where the optimal value of a tuning parameter, say α, needs to be found, is given by

$$CV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\mathcal{K}(i)}(\mathbf{x}_i, \alpha)).$$

In this expression, $\mathcal{K}(i)$ is an indexing function indicating the partition k to which observation i is allocated and $\hat{f}^{-k}(\mathbf{x}_i, \alpha)$ denotes the i^{th} predicted response obtained from the model fit on the training set where the k^{th} part of the data were removed.

A carefully selected subset of candidate values for α is considered. These α's can be plotted against $CV(\hat{f}, \alpha)$. The α corresponding to the minimum estimated validation error will be selected for the final model.

The U-shaped pattern in Figure 1.3, as well as the cause of overfitting and underfitting, can be explained by what is called the bias-variance trade-off. The argument stems from a decomposition of the squared error that is derived below. Firstly, the expected error between the true response and the estimate provided by the model, over the training sample, can be decomposed into a reducible error term and an irreducible error term. This decomposition, which stems from Equation 1.3, is shown below:

$$\begin{aligned} E_{\mathcal{T}}[(Y - \hat{f}(\mathbf{x}))^2] &= E_{\mathcal{T}}[f(\mathbf{x}) - \hat{f}(\mathbf{x})]^2 + E_{\varepsilon}(\varepsilon)^2 \\ &= E_{\mathcal{T}}[f(\mathbf{x}) - \hat{f}(\mathbf{x})]^2 + Var_{\varepsilon}(\varepsilon). \end{aligned}$$

The first term relates to the error stemming from the process of estimating the true underlying relationship by a model built on training data. The second term relates to the inherent noise in the data, which cannot be reduced.

Attempts to improve the accuracy of the model therefore focus on the first term. This can be further decomposed as follows:

$$\begin{aligned} &E_{\mathcal{T}}[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2] \\ &= E_{\mathcal{T}}\big[\{f(\mathbf{x}) - E_{\mathcal{T}}(\hat{f}(\mathbf{x}))\} + \{E_{\mathcal{T}}(\hat{f}(\mathbf{x})) - \hat{f}(\mathbf{x})\}\big]^2 \\ &= \big(f(\mathbf{x}) - E_{\mathcal{T}}[\hat{f}(\mathbf{x})]\big)^2 + 2\big(f(\mathbf{x}) - E_{\mathcal{T}}[\hat{f}(\mathbf{x})]\big) E_{\mathcal{T}}\big[E_{\mathcal{T}}[\hat{f}(\mathbf{x})] - \hat{f}(\mathbf{x})\big] \\ &\quad + E_{\mathcal{T}}\big[E_{\mathcal{T}}[\hat{f}(\mathbf{x})] - \hat{f}(\mathbf{x})\big]^2 \\ &= \big(f(\mathbf{x}) - E_{\mathcal{T}}[\hat{f}(\mathbf{x})]\big)^2 + E_{\mathcal{T}}\big[E_{\mathcal{T}}[\hat{f}(\mathbf{x})] - \hat{f}(\mathbf{x})\big]^2 \\ &= [Bias(\hat{f}(\mathbf{x}))]^2 + Var(\hat{f}(\mathbf{x})). \end{aligned}$$

Stellenbosch University https://scholar.sun.ac.za

The error has therefore been decomposed into a bias term and a variance term. The bias term refers to the average loss of accuracy that is incurred when a complex process is modelled by a simple model. The variance term refers to the extent to which the model varies when it is estimated on a different training set.

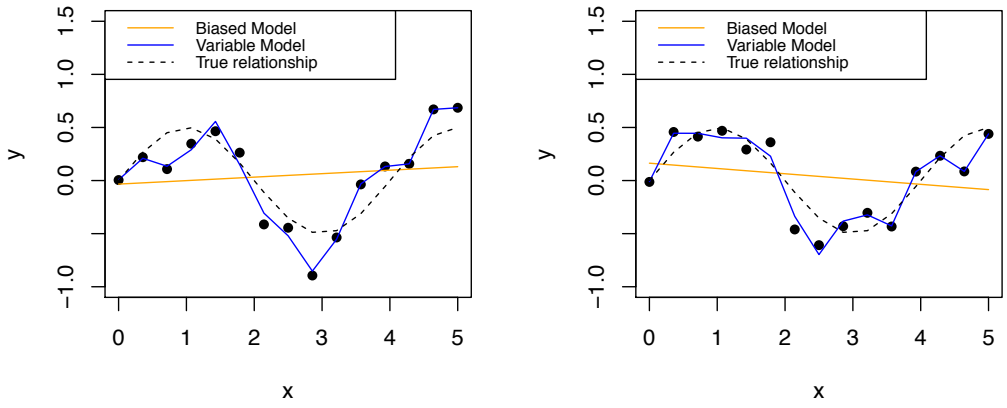

Figure 1.4: Behaviour of models with high bias and high variance respectively.

To illustrate the behaviour of these two terms, two models are compared in Figure 1.4. The two sets of sample points in the left and right panels are both generated by the relationship $y = \frac{1}{2}\sin(x \times \frac{\pi}{2}) + \varepsilon$, where $\varepsilon \sim N(0, 0.25)$. The dashed line shows the relationship when ignoring the noise. Model 1 is a linear model and it is clear that the model underfits the data; however, the model is very stable and exhibits a low variance. The linear model is too simplistic to capture the complex relationship and therefore the bias is large. Model 2 is a polynomial of degree 50, giving it increased flexibility compared to Model 1. However, this model changes significantly between the two samples, giving rise to a high model variance. Model 2 seems to be overfitting, since the noise components are seemingly also described by the model.

To minimise the model error, the aim is to minimise both the bias and the variance components. This turns out to be a difficult task, since these terms are often in competition with each other. If we were to decrease the bias in Model 1 by fitting a slightly more complex model, then the increased flexibility would also give rise to a higher variance. The opposite is true for Model 2. This is commonly referred to as the bias-variance trade-off, and is depicted in Figure 1.5. Ultimately the relative rate of change of squared bias and of variance will determine whether the expected squared error is reduced or not.

Stellenbosch University https://scholar.sun.ac.za

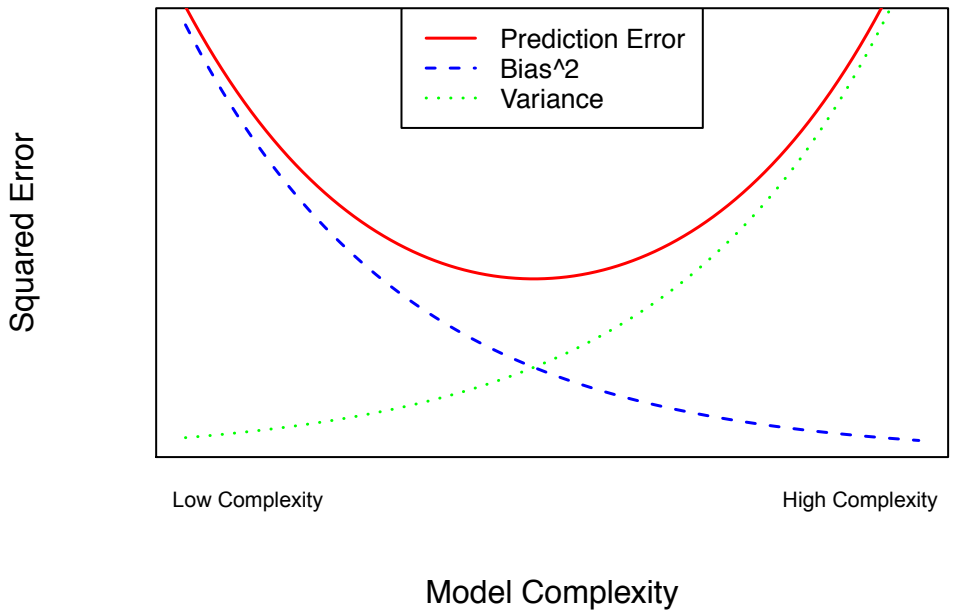

Figure 1.5: The bias-variance trade-off.

For classification, however, using the misclassification rate (also referred to as 0-1 loss), this same argument does not hold. Friedman (1997) shows that when an observation is already correctly classified, then the bias term is negative and therefore it decreases the prediction error irrespective of its size. If the prediction is incorrect, i.e. on the wrong side of the decision boundary, then an increase in variance can actually decrease the prediction error. This framework is therefore not recommended for use in the classification setting and the expected loss should be focussed on directly instead.

Overfitting is not only impacted by the model complexity, but also by the number of observations in the training sample relative to the number of input variables in the data. With fewer training sample points in an input space of a fixed dimensionality, distinguishing between the irreducible error and the model error becomes more difficult, as the model fit becomes proportionally more reliant on each individual point. Overfitting is therefore more likely, especially in the case where local methods, such as KNN, are used.

Increased dimensionality of data is a clear trend that has been observed over the past few decades (Donoho, 2000). Traditionally, variables possibly influencing a response were identified before an investigation was performed and data were captured laboriously in order for the hypothesis to be tested or the data to be analysed. Recently however, data have become much easier to capture, collect, analyse and share, so that it is now possible to build an

Stellenbosch University https://scholar.sun.ac.za

investigation around the available data. For this reason, it is felt that capturing more variables can be of great potential value in discovering previously unthought of relationships. The increased dimensionality of the data, however, frequently results in a decrease in prediction accuracy, as well as in the interpretability of the model.

Poor generalisation performance of models in high-dimensional cases, where $p > N$, can be explained by what has been termed *the curse of dimensionality* (Bellman, 1961). Three ways in which this phenomenon manifests are briefly discussed below.

The first way in which it is manifested, is that points become sparse in high dimensions. The distance between any sample point and the nearest other point increases. This means that there are fewer points available in the local vicinity of a new target point from which to infer a response value. Relying more heavily on a smaller number of local observations makes the model prone to overfitting, which leads to poor generalisation.

The second way is linked to the above idea of sparsity in high dimensions. It concerns the sample density, which can be described as the percentage of sample points to be found in a region of specified size in the input space. The number of points needed to support a constant sample density in a region grows exponentially as dimensionality increases. This number can quickly become unrealistically large as more input variables are added to the dataset. Or, stated differently, if the number of sample points is not increased, then, when considering a fixed proportion of the data around a sample point, the area or volume in the input space under consideration needs to increase exponentially relative to the increase in dimension. This implies that points in the neighbourhood of the target point are less similar to it, since they lie further away. Therefore, using the average response of the neighbouring points to predict an outcome for a new point will result in averaging over a large portion of the input space. Such a model will be quite inflexible: being constant in some regions, with discontinuous jumps between the regions.

The third manifestation of the curse of dimensionality is that points lie nearer to the boundary of the input space than to its centre. This is problematic since prediction is difficult near the boundary of the input space. For example, predicting a response for a target point near the boundary by considering the neighbouring points, implies that the prediction will be biased towards points in the interior of the input space. To account for this, special assumptions need to be made for dealing with points near or beyond the input space boundary. Such assumptions may not result in an accurate or sufficiently complex model, especially since most points will tend to fall in this region in higher dimensions.

Stellenbosch University https://scholar.sun.ac.za

It is clear from the above arguments that situations where $p > N$ are particularly challenging. If an infinite amount of data were available, then the curse would be stripped of its powers. However, since the amount of data remains finite, methods to combat the curse of dimensionality are essential. The three main approaches are to restrict the model by assuming a structure (i.e. by using a parametric model), to perform regularisation and to use dimension reduction techniques. In the next chapter these approaches are discussed in detail.

The predictive capability of a model is not the only element that should be considered when selecting a model. The form of the model itself may enhance the understanding of the true underlying relationship between the input and output, which can therefore be used for inference. Aside from the overfitting argument, fewer variables included in a model may also be preferable purely from an interpretational point of view. Since certain types of predictive models are more amenable to inference, the purpose of the model in this regard should be considered in the model selection phase.

1.1.2 Unsupervised Learning and Reinforcement Learning

The other branch of statistical learning, namely unsupervised learning, does not try to predict an outcome using input variables, but rather attempts to find patterns and similarities within the input variables. The goal is therefore not to determine a specific value or class, but rather to gain a better understanding of the source or subject of the data. Since there is no metric to measure performance by, this branch is less well defined than supervised learning. Cluster analysis and principal component analysis are examples of unsupervised techniques. Although the main focus of the research will be within the former category (supervised learning), principal component analysis forms a key element of some of these predictive techniques and will therefore be discussed in detail in Chapter 2.

Reinforcement learning is mainly used in robotics and is the newest addition to the machine learning realm. Reinforcement learning techniques do not learn from labelled data, but rather from positive and negative feedback based on their interactions within an environment. The goal is to maximise some reward function. Reinforcement learning is beyond the scope of this document.

1.2 Motivation and Thesis Objectives

The motivation for this study is to gain a better understanding of preconditioning as a method for variable selection in a classification setting. Of specific

Stellenbosch University https://scholar.sun.ac.za

interest is how well the method performs compared to other commonly used variable selection methods, whether it performs well in situations where the data are noisy, where the class distributions overlap and where the number of variables is large compared to the number of observations.

The objectives of the study are to provide a summary of the current literature on the topic, to test the usefulness of the technique in various scenarios by means of a simulation study and to see whether the results from the simulation study extend to real world problems.

1.3 Outline

In the next chapter, Chapter 2, methods to reduce dimensionality and control model variance are discussed in detail. This includes discussions on variable selection techniques, shrinkage methods, feature selection methods, as well as preconditioning.

In Chapter 3, the methodology and results of a simulation study are presented. In Chapter 4, the techniques established in Chapter 3 are applied to real-world datasets and the results are discussed. In Chapter 5, final results and conclusions are discussed. In Appendix A the variances of the results of the simulation studies are presented and discussed. Appendix B provides the R-code used to generate the results of Chapters 3 and 4.

1.4 Concluding Remarks

This chapter introduced the topic of statistical learning theory, with its subdivisions, supervised learning and unsupervised learning. Important concepts such as loss functions and prediction error were described for both regression and classification settings, as well as methods to obtain the parameter estimates. Furthermore, overfitting, the bias-variance trade-off and the curse of dimensionality were defined and examples were provided.

The thesis motivation, objectives and outline were also stated.

In the next chapter, specific ways in which the curse of dimensionality can be managed are discussed in detail.

Stellenbosch University https://scholar.sun.ac.za

Chapter 2

Dimension Reduction and Regularisation

In the previous chapter it was mentioned that regularisation and dimension reduction can be used to combat overfitting and the curse of dimensionality. The curse of dimensionality stems from the presence of a large number of input variables and an intuitive way of alleviating the problems associated with the phenomenon, is to reduce the number of variables that are used within the final model. This is the purpose of dimension reduction. Dimension reduction techniques can be divided into two parts, namely variable selection and feature selection. Both approaches reduce the number of parameters in the final model, but the distinction lies therein that variable selection achieves this by selecting only a subset of the original input variables, whereas feature selection achieves this by basing the model on a smaller number of transformed input variables. Another approach to control the curse of dimensionality is to impose restrictions on the model that will reduce model complexity, and thus variance. This can be done by regularisation methods.

The present chapter is structured as follows. Section 2.1 concerns variable selection techniques and includes discussions on best subset selection, forward- and backward stepwise selection, forward-stagewise regression and sure independence screening. In Section 2.2, ridge regression is discussed as a shrinkage method. In Section 2.3, the lasso and nearest shrunken centroids are discussed as mixtures of shrinkage and selection methods. The topic of Section 2.4 is feature selection and within this section, principal component analysis (PCA), supervised PCA and Y-aware PCA are discussed. Finally, the chapter concludes with Section 2.5 presenting a discussion of a technique called preconditioning.

Stellenbosch University https://scholar.sun.ac.za

2.1 Variable Selection

The field of variable selection attempts to determine which of the input variables in the original dataset will have the most predictive power, and then excludes all the other variables in order to reduce the dimensionality of the problem. Distinguishing between important variables and unimportant variables, however, is not a straightforward task when only a finite sample of data is provided. Some variables may appear to influence the response significantly, but this could be a coincidental result of random variation. Other variables that truly are significant may be disguised by noisy variables that happen to have a strong correlation with the significant variable based on the sample drawn. It is also possible that a truly significant variable seems to be irrelevant based purely on the sample of data selected. A trivial example of this is when a predictor is built to determine the body mass index (BMI) of a person, which is defined to be a function of height and weight. If a model is trained on a sample where all the subjects in the sample are of the same height, then height will not be included as a relevant variable in the model. Without the prior knowledge of the importance of height in the calculation of BMI, the model would be too simplistic. In general, it may not be possible to detect important variables when the sample is chosen poorly; however, by using stratified sampling, this issue may be avoided. These difficulties should be kept in mind when considering the uncertainty within variable importance measures.

In practice, data are sometimes messy and an initial screening of suitable variables based on completeness and variance characteristics, could provide some reduction in the value of p. Variables with all missing, or many missing values will not contribute much to the model and can therefore be excluded from the outset. Similarly, variables that have zero, or near-zero, variance are essentially constant and can be excluded so that their fixed contribution can be absorbed into the intercept term of the model.

If the dataset is of a good quality, the above steps will have little impact on the analysis and one can commence with more sophisticated methods for conducting variable selection. The variable selection methods below are described within the context of a linear regression model, but they are also applicable to other model types.

2.1.1 Best Subset Selection

This method proceeds as follows. For every combination of k predictors, where $k \in \{1, 2, ..., p\}$, the error on the training set is computed. This implies that 2^p models are fitted. For each value of k, the combination of k predictors resulting in the model with the minimum error is noted. From this smaller subset of

Stellenbosch University https://scholar.sun.ac.za

p models, the cross-validated prediction errors are calculated and the final selection will be the model corresponding to the minimum cross-validation error.

The model obtaining the overall lowest error on the training set will always be the one containing all the variables, as the training error decreases monotonically as k increases. For this reason, cross-validation is used to obtain an estimate of the generalisation error in the last step. If this is minimised, it is possible that a $k < p$ could be selected that results in dimension reduction.

Since the number of variable combinations grows rapidly as p increases, this technique is only feasible for $p \leq 40$ (Hastie *et al.*, 2009, p. 58). It should be noted that there is no guarantee that the best subsets for different values of k will be nested solutions. In other words, the best subset with i variables will not necessarily include all the variables from the best subset of $i-1$ variables. This is another drawback of best subset selection, since this phenomenon is not easy to explain.

2.1.2 Forward- and Backward-Stepwise Selection

Since best subset selection becomes infeasible for datasets with more than 40 predictors, this selection method cannot always be used. Other techniques, which are greedier by nature, have therefore been developed to handle data where p exceeds 40. The term greedy refers to the characteristic that a method will attempt to make a locally optimal choice at each stage in the hope of obtaining a high quality solution/model. This behaviour results in nested models in this case, therefore addressing another drawback of best subset selection. The drawback of greedy models is that the final solution may not be globally optimal.

Forward-stepwise selection is a selection method which is greedy in nature. It starts with an intercept term and then adds the predictor term that will improve the fit most. The algorithm will continue adding the next predictor that will improve the fit the most until the chosen subset size is reached. As with best subset selection, the subset with the lowest training error will be the subset containing all the predictors. The optimal subset-size hyper-parameter, k, could therefore be estimated by cross-validation. Forward stepwise selection can be used for any number of observations and predictors.

Backward-stepwise selection is a similar technique, but works in the reverse direction: it starts with the full model and sequentially deletes the predictor that has the least impact on the fit. The measure used for determining the impact on the fit is the Z-score, defined as the coefficient estimate, divided by its standard error. The predictor with the lowest Z-score is eliminated at every step. Backward-stepwise selection can only be used when the number of

Stellenbosch University https://scholar.sun.ac.za

observations exceeds the number of predictors.

Both these methods' performances are often comparable to that of best subset selection, but they have the added benefit of being much less computationally intensive.

2.1.3 Forward-Stagewise Regression

Whereas the selection methods described so far have been a discrete process of either dropping or keeping any specific variable, forward-stagewise regression is a more continuous method. It starts with an intercept term and coefficients equal to zero for all the predictors. It then finds the predictor which is most correlated with the current residual and computes a simple linear regression coefficient of the residual on this predictor. The residual begins as the value of y and in subsequent steps is the residual from the current model. The value of the coefficient is added to the current value of the corresponding coefficient, i.e. $\beta_j^{new} = \beta_j + \langle \mathbf{x}_j, \mathbf{r} \rangle$, where $\mathbf{x}_j$ has been standardised and $\mathbf{r}$ is the residual. At each iteration, only the coefficient of the variable most correlated to the residual is adjusted. This is unlike forward-stepwise, where the entire least squares model is updated following the inclusion of an additional variable. The coefficients are enlarged in each successive step until none of the variables have correlation with the residuals. The final model will be the least-squares fit. The algorithm typically takes more than p steps to reach this solution, because of the incremental way in which the coefficients are adjusted. The aim is to stop the procedure before all the predictors have been added, so a suitable $k \leq p$ could be selected beforehand, or determined by cross-validation.

This method implements slow fitting, since coefficients are adjusted one at a time, and only by the amount equal to the correlation of the variable with the residual (provided X is standardised). By taking many small steps towards the full least squares solution, a more thorough examination of the surface of the loss function (on a validation set) is performed in search of a local minimum. In high-dimensional problems, where overfitting is likely, slow fitting models therefore can be beneficial. This characteristic, together with the computational efficiency of only fitting a univariate model at each iteration, makes forward-stagewise regression well suited to high-dimensional problems.

An extension of the forward-stagewise algorithm is a method that is even more slow fitting, namely incremental forward-stagewise regression. This algorithm commences as above, but instead of updating the coefficients with the full correlation coefficient, they are adjusted only by a small amount, $\epsilon > 0$, in the direction of the correlation, i.e. $\beta_j^{new} = \beta_j + \epsilon \cdot \text{sign}[\langle \mathbf{x}_j, \mathbf{r} \rangle]$. This method is closely linked to the least angle regression (LAR) algorithm that is referred to in the discussion on the lasso in Section 2.3.1 (Hastie *et al.*, 2007).

Stellenbosch University https://scholar.sun.ac.za

2.1.4 Sure Independence Screening

When the number of variables is very large, certain selection methods can be computationally expensive and some may even fail. Sure Independence Screening (SIS) is a variable selection technique, proposed by Fan and Lv (2008), that is particularly well suited to very high-dimensional data. The term *sure screening* refers to the property that, with probability tending to one, all the variables of importance survive the screening process. It is introduced as an initial screening process which reduces p to below N. The idea is that other selection techniques can then be successfully utilised on the remaining variables to reduce the dimensionality further, if required.

Suppose X is standardised so that each of its columns has a mean of zero and a variance of one and let $\mathcal{M}_* = \{1 \leq j \leq p : \beta_j \neq 0\}$ denote the true underlying model. Let $\omega = X^T\mathbf{y}$ be the coefficient vector obtained by univariate regression of Y on each of the inputs. When X and $\mathbf{y}$ are both standardised then ω is a vector of marginal correlations between the input variables and the response.

Now suppose the correlations are sorted in decreasing order of absolute size, then for any scaling factor, $\gamma \in (0, 1)$, a new sub-model can be defined:

$$\mathcal{M}_\gamma = \{1 \leq j \leq p : |\omega_j| \text{ is among the first } [\gamma N]\text{-largest of all}\}$$

The notation $[\gamma N]$ refers to the integer part of γN. The submodel $\mathcal{M}_\gamma$ is therefore of size $d = [\gamma N] < N$, which can represent a significant reduction in dimensionality, especially when p is much larger than N.

An alternative way to formulate the above is to define a reduced input variable matrix, $X_\theta : N \times r$ where $r < p$, as the matrix consisting only of the columns of X for which $|\omega_j| > \theta$ for $j = 1, 2, \ldots, p$, where θ is the threshold value (Bair *et al.*, 2006). The threshold, θ (or γ in the first formulation) can be found by cross-validation. This alternative formulation differs from the initial formulation only in that, unlike d, r is not restricted to be strictly less than N.

The principle behind SIS applies even if the measure used for estimating variable importance is chosen differently. Bair *et al.* (2006) recommend the use of the following score statistic for the general case (i.e. either a regression problem or a classification problem):

$$s_j = \frac{U_j(0)^2}{I_j(0)},\ j \in \{1, 2, \cdots, p\},$$

where the numerator is defined as $U_j(\beta_0) = \frac{\partial l_j(\beta)}{\partial \beta}|_{\beta=\beta_0}$, and the denominator is defined as $I_j(\beta_0) = -\frac{\partial^2 l_j(\beta)}{\partial \beta^2}|_{\beta=\beta_0}$, with $l_j(\beta)$ the log likelihood function for the

Stellenbosch University https://scholar.sun.ac.za

univariate regression on the j-th covariate. For a Gaussian log-likelihood, this quantity is equivalent to the standardised univariate regression coefficient. In the binary classification context, a score statistic can be derived as follows.

Consider a binary classification scenario where $Y \in \{0, 1\}$. Assume that the underlying probability distribution is a Bernoulli distribution. The likelihood function for input X_j is given by

$$\begin{aligned} L(\mathbf{y}, p(\mathbf{x}_j)) &= \prod_{i=1}^{N} p(x_{ij})^{y_i}(1 - p(x_{ij}))^{(1-y_i)} \\ &= \prod_{i=1}^{N} \left[\frac{p(x_{ij})}{1 - p(x_{ij})}\right]^{y_i} [1 - p(x_{ij})] \\ log(L(\mathbf{y}, p(\mathbf{x}_j))) &= \sum_{i=1}^{N} \left\{ y_i log \left[\frac{p(x_{ij})}{1 - p(x_{ij})}\right] + log[1 - p(x_{ij})] \right\}. \end{aligned}$$

Suppose logistic regression is used to model the probability of a success, then $log\left[\frac{p(x_{ij})}{1-p(x_{ij})}\right] = \beta_{0j} + \beta_{1j}x_{ij}$ and consequently $p(x_{ij}) = \frac{e^{\beta_{0j}+\beta_{1j}x_{ij}}}{1+e^{\beta_{0j}+\beta_{1j}x_{ij}}}$. By substitution the log-likelihood can therefore be written as:

$$\begin{aligned} log(L(\mathbf{y}, p(\mathbf{x}_j))) &= \sum_{i=1}^{N} \left\{ y_i \left[\beta_{0j} + \beta_{1j}x_{ij}\right] + log\left[\frac{1}{1 + e^{\beta_{0j}+\beta_{1j}x_{ij}}}\right] \right\} \\ &= \sum_{i=1}^{N} \left\{ y_i \left[\beta_{0j} + \beta_{1j}x_{ij}\right] - log\left[1 + e^{\beta_{0j}+\beta_{1j}x_{ij}}\right] \right\}. \end{aligned}$$

For notational simplicity, let $l_j = log(L(\mathbf{y}, p(\mathbf{x}_j)))$. Now taking the first derivative in terms of β_{1j} and evaluating it at zero, yields the following expression for the numerator of the score statistic:

$$\frac{\partial l_j}{\partial \beta_{1j}} = \sum_{i=1}^{N} \left\{ x_{ij}y_i - \frac{x_{ij}e^{\beta_{0j}+\beta_{1j}x_{ij}}}{1 + e^{\beta_{0j}+\beta_{1j}x_{ij}}} \right\}$$

$$U_j(0) = \frac{\partial l_j}{\partial \beta_{1j}}|_{\beta=\mathbf{0}} = \sum_{i=1}^{N} \left[x_{ij}y_i - \frac{x_{ij}}{2}\right] = \sum_{i=1}^{N} x_{ij}(y_i - \frac{1}{2}). \tag{2.1}$$

Taking the second derivative of l_j in terms of β_{1j} and evaluating $\beta = [\beta_{0j}\ \beta_{1j}]$ at zero results in the following expression for the denominator of the score

Stellenbosch University https://scholar.sun.ac.za

statistic:

$$\begin{aligned}\frac{\partial^2 l_j}{\partial \beta_{1j}^2} &= -\sum_{i=1}^{N}\left\{\frac{x_{ij}^2 e^{\beta_{0j}+\beta_{1j}x_{ij}}[1+e^{\beta_{0j}+\beta_{1j}x_{ij}}]-[x_{ij}e^{\beta_{0j}+\beta_{1j}x_{ij}}]^2}{[1+e^{\beta_{0j}+\beta_{1j}x_{ij}}]^2}\right\}\\ &= -\sum_{i=1}^{N}\frac{x_{ij}^2 e^{\beta_{0j}+\beta_{1j}x_{ij}}}{[1+e^{\beta_{0j}+\beta_{1j}x_{ij}}]^2}\\ I_j(0) &= -\frac{\partial^2 l_j}{\partial \beta_{1j}^2}|_{\beta=\mathbf{0}} = \frac{\sum_{i=1}^{N} x_{ij}^2}{(1+1)^2} = \frac{1}{4}\sum_{i=1}^{N} x_{ij}^2.\end{aligned}$$

The final score statistic takes on the following form:

$$\begin{aligned}s_j &= \frac{U_j(0)^2}{I_j(0)}\\ &= \frac{\left[\sum_{i=1}^{N} x_{ij}\left(y_i-\frac{1}{2}\right)\right]^2}{\frac{1}{4}\sum_{i=1}^{N} x_{ij}^2}.\end{aligned}$$

The numerator is N^2 times the squared covariance of $\mathbf{x}$ and $\mathbf{y}$, since, when $\mathbf{x}$ is centred and each element $y_i \in \{0,1\}$ and the dataset is balanced, the covariance can be simplified as follows:

$$Cov(\mathbf{x},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y}) = \frac{1}{N}\sum_{i=1}^{N} x_i\left(y_i-\frac{1}{2}\right).$$

Therefore, for this special case,

$$\left[\sum_{i=1}^{N} x_i\left(y_i-\frac{1}{2}\right)\right]^2 \times \frac{N^2}{N^2} = N^2[Cov(\mathbf{x},\mathbf{y})]^2.$$

When Y has a Bernoulli distribution and it is assumed that the population data are balanced so that $p=\frac{1}{2}$, then the variance of Y is $p(1-p)=\frac{1}{4}$. Therefore the denominator can be written as $N \times Var(Y)Var(\mathbf{x})$ when $\mathbf{x}$ is centred.

The final score statistic therefore takes the form,

$$\begin{aligned}s_j &= N \cdot \frac{[Cov(\mathbf{x}_j,\mathbf{y})]^2}{Var(\mathbf{x}_j)Var(\mathbf{y})}\\ &= N \cdot [corr(\mathbf{x}_j,\mathbf{y})]^2.\end{aligned}$$

Since the constant in front will be identical for all $j=1,2,\cdots,p$ and only the relative sizes of the scores are of interest when SIS is performed, the factor N can be ignored. This leads to the conclusion that when the proportions of

Stellenbosch University https://scholar.sun.ac.za

successes and failures in the data are equal, then the squared univariate correlation between the variables and the response can be used for the purpose of screening.

It is also interesting to note that in the case where $Y \in \{0, 1\}$, Equation (2.1) can be rewritten as

$$\sum_{i=1}^{N} x_{ij}\left(y_i - \frac{1}{2}\right) = \frac{1}{2}\left(\sum_{\{i:y_i=1\}}^{N} x_{ij} - \sum_{\{i:y_i=0\}}^{N} x_{ij}\right),$$

and therefore the score statistic can also be rewritten as

$$s_j = \frac{\left(\sum_{\{i:y_i=1\}} x_{ij} - \sum_{\{i:y_i=0\}} x_{ij}\right)^2}{\sum_{i=1}^{N} x_{ij}^2}.$$

In this form, the numerator can be seen as a measure of the distance, or degree of separation, between the variable values associated with the first class and the variable values associated with the second class. The denominator can be thought of as a type of standardisation to make the score scale invariant. Both of the forms in which the score statistic can be written have interpretations that are intuitively appealing.

2.2 Shrinkage Methods

Selection methods drop variables in a discrete manner and therefore may display large variance in the model across different training datasets. Another method of controlling the degree of overfitting is to use shrinkage methods. These methods continuously shrink coefficients towards zero by the use of a penalty term. The reasoning behind this is that larger coefficient sizes will result in a larger change when the (sample) input variable is only slightly changed. This type of behaviour is typical of overfitting, leading to a higher variance model.

Consider a general formulation of the optimisation problem to be solved in a regression scenario, i.e.

$$\min_{\beta}\left\{\sum_{i=1}^{N} L\left(y_i, f_\beta(\mathbf{x_i})\right)\right\}.$$

In this expression $f_\beta(\cdot)$ is a function depending on the parameter vector β, and $L(\cdot, \cdot)$ is a loss function. If p is large, or if the predictor variables are highly correlated, solving this optimisation problem may yield unstable results. This

Stellenbosch University https://scholar.sun.ac.za

problem can be addressed by considering instead the modified (regularised) optimisation problem

$$\min_\beta \left\{ \sum_{i=1}^{N} L\left(y_i, f_\beta(\mathbf{x_i})\right) + \lambda J(\beta) \right\}.$$

In this expression $J(\beta)$ is a (positive) penalty term, while λ is a complexity parameter which balances lack-of-fit (reflected in the loss function) and complexity of $f_\beta(\cdot)$ (reflected in $J(\beta)$). The value of λ is typically determined by cross-validation. By considering the modified instead of the original optimisation problem, the solution $\hat{\beta}(\lambda)$ will typically have smaller absolute components than $\hat{\beta}(0)$, the solution to the original optimisation problem. This can largely eliminate the instability in the values of the components of $\hat{\beta}(0)$.

Ridge regression is a specific instance of this general formulation and is discussed below.

2.2.1 Ridge Regression

Ridge regression penalises large coefficients in a linear regression model by minimising a penalised sum of squared error, set out below:

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \ ; \ \text{where } \lambda \geq 0.$$

The quantity λ is known as the penalty parameter. The larger the value of λ, the closer the estimated coefficients will tend to zero. Since λ is multiplied by $\sum_{j=1}^{p} \beta_j^2$, larger coefficients will receive a higher penalty. By minimising the penalised sum of squared errors, smaller absolute coefficients, that move away from the least squares solution, are given preference.

A closed form expression for the solution to this modified optimisation problem is available, similar to Equation 1.2,

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}. \tag{2.2}$$

The reader is referred to Hastie *et al.* (2009, p. 64) or Murphy (2012, p. 225) for a more in-depth discussion of ridge regression.

In general, penalising parameters in any way is a form of regularisation. The penalty term in the ridge regression optimisation equation is often referred to as L_2-regularisation and can be applied to other parametric models, such as logistic regression, as well. The 2 relates to the exponent of the β_j's in the penalty term. In general, L_p-regularisation has a penalty term $\lambda \sum_{j=1}^{p} |\beta_j|^p$. Where a closed form solution is not available, optimisation algorithms can be used to approximate the penalised estimated model coefficients.

Stellenbosch University https://scholar.sun.ac.za

2.3 Shrinkage and Selection Methods

Selection methods have the advantage of reducing the dimensionality of a problem, whereas shrinkage methods have the advantage of reducing the variance of the model (in exchange for a small increase in bias). Some techniques, such as the lasso and nearest shrunken centroids, are able to combine these two approaches in order to benefit from the advantages of both shrinkage and selection. These two techniques are discussed below.

2.3.1 Lasso Regression

The lasso is a regularisation method similar in form to ridge regression. Unlike ridge, however, the lasso has the advantage of being able to shrink coefficients to be exactly zero. This property enables the lasso to achieve both shrinkage and selection. The lasso coefficient estimates are given by the following equation:

$$\hat{\beta}^{lasso} = \text{argmin}_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \; ; \; \text{where } \lambda \geq 0. \tag{2.3}$$

The difference between the ridge penalty term and the lasso penalty term is that the penalty of the lasso is proportional to $\sum_{j=1}^{p} |\beta_j|$ instead of $\sum_{j=1}^{p} \beta_j^2$. This difference allows the lasso coefficient estimates to be shrunk to exactly zero, for a large enough λ. In this sense, the lasso is not only a shrinkage technique, but also a technique which performs selection. This alteration also makes the solution non-linear in y_i and no closed form solution is available.

The solution to the lasso optimisation problem can be efficiently computed by the least angle regression (LAR) algorithm. This method is closely related to the incremental forward-stagewise algorithm discussed in Section 2.1.3. Instead of adjusting a coefficient by the fixed step size ϵ, it is moved towards its least squares fit only until another variable has as much correlation with the residual. This next variable is then added and the coefficients of all the variables included in the model up to that point are then moved in the direction of their joint least squares fit on the residual. The coefficient sizes form piecewise linear functions as the algorithm progresses, with knots at the points where a new variable enters the model. If allowed to run until none of the predictors are correlated with the response, the solution will be the least squares fit. The property of piecewise linearity of the algorithm together with the covariance of the input variables can be used to calculate the step lengths precisely. When a variable is dropped from the model if its coefficient crosses zero, then the solution given by LAR is equivalent to the lasso solution. LAR takes $\min(N-1, p)$ steps to reach the least squares solution and is therefore

Stellenbosch University https://scholar.sun.ac.za

especially efficient when $p \gg N$. The reader is referred to Efron *et al.* (2004) for further computational details.

Alternatively, an iterative approach known as pathwise coordinate optimisation, can be used. This technique is based on simple coordinate descent and is briefly summarised below.

For a fixed penalty parameter λ, the algorithm optimises each coefficient one at a time, while holding the values of the other coefficients fixed. Let $\tilde{\beta}(\lambda) = [\tilde{\beta}_1(\lambda), \tilde{\beta}_2(\lambda), \cdots, \tilde{\beta}_p(\lambda)]^T$ where the k-th element is the current value of the estimate of β_k, at a given value of λ. The optimisation can be done by rewriting Equation 2.3 and dropping the intercept term as follows:

$$R(\tilde{\beta}(\lambda), \beta_j) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \sum_{k\neq j}\tilde{\beta}_k(\lambda)x_{ik} - \beta_j x_{ij})^2 + \lambda\sum_{k\neq j}|\tilde{\beta}_k(\lambda)| + \lambda|\beta_j|.$$

This can then be viewed as a univariate lasso problem where the response values are the partial residuals $y_i - \sum_{k\neq j}\tilde{\beta}_k(\lambda)x_{ik}$ and the predictor values are x_{ij}. Let $\tilde{y_i}^{(j)} = \sum_{k\neq j}\tilde{\beta}_k(\lambda)x_{ik}$, then the simple regression coefficient of the partial residual $y_i - \tilde{y_i}^{(j)}$ on the standardised variable $\mathbf{x}_j$ can be written as $\langle \mathbf{x}_j, \mathbf{y} - \tilde{\mathbf{y}}^{(j)}\rangle = \sum_{i=1}^{N} x_{ij}(y_i - \tilde{y_i}^{(j)})$. This univariate lasso problem has an explicit solution, resulting in the update below:

$$\tilde{\beta}_j(\lambda) \leftarrow \text{ sign}\left(\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y_i}^{(j)})\right)\left(\left|\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y_i}^{(j)})\right| - \lambda\right)_+ .$$

When applying the above for $j = 1, 2, \cdots, p$ and cycling through these iteratively until convergence, the lasso estimate $\hat{\beta}(\lambda)$ is obtained for a given λ.

Implementing the penalty term in the lasso coefficient estimate is known as L_1-regularisation and, as in the case of ridge, is not only limited to application in linear regression. It can be applied to other parametric models. In this thesis, penalised logistic regression (PLR) refers to a model which regularises the coefficients of a logistic model by use of the L_1-penalty term in the optimisation function. The solution of such models can be found by the LAR algorithm or by using pathwise coordinate optimisation.

2.3.2 Nearest Shrunken Centroids

Nearest shrunken centroids (NSC) is a classifier that also combines shrinkage and selection. The model is explained by linking it to linear discriminant analysis (LDA) and related methods. These topics are briefly described below and thereafter NSC will be described.

Stellenbosch University https://scholar.sun.ac.za

Linear Discriminant Analysis

Linear discriminant analysis is a classification technique that pre-dates logistic regression. As in logistic regression, suppose one would like to model the posterior class probabilities, $P(Y = k|\mathbf{X} = \mathbf{x})$, in order to classify observations to a particular class. If $f_k(\mathbf{x})$ denotes the class-conditional density of $\mathbf{X}$ in class $Y = k$ and π_k denotes the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$, then by applying Bayes' theorem, the posterior probabilities can be expressed as

$$P(Y = k|\mathbf{X} = \mathbf{x}) = \frac{f_k(\mathbf{x})\pi_k}{\sum_{i=1}^{K} f_i(\mathbf{x})\pi_i}.$$

Unlike logistic regression, an assumption is made regarding the form of the probability density functions of the input variables. In LDA it is assumed that each class density is a multivariate normal distribution, taking on the following form:

$$\begin{aligned} & X|Y = k \sim N_p(\mu_k; \Sigma_k) \\ \implies & f_k(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)}. \end{aligned}$$

Assuming that the classes have a common covariance matrix $\Sigma_k = \Sigma$, for classes k and l the decision boundary which separates predictions to the one class or the other, lies where the posterior probabilities, $P(Y = k|\mathbf{X} = \mathbf{x})$ and $P(Y = l|\mathbf{X} = \mathbf{x})$, are equal. The algorithm will therefore classify to the class with the highest posterior probability. From the above two equations, it follows that class k will be selected instead of class l when

$$\begin{aligned} & P(Y = k|\mathbf{X} = \mathbf{x}) > P(Y = l|\mathbf{X} = \mathbf{x}) \\ \iff & \frac{f_k(\mathbf{x})\pi_k}{\sum_{i=1}^{K} f_i(\mathbf{x})\pi_i} > \frac{f_l(\mathbf{x})\pi_l}{\sum_{i=1}^{K} f_i(\mathbf{x})\pi_i} \\ \iff & f_k(\mathbf{x})\pi_k > f_l(\mathbf{x})\pi_l \\ \iff & log(f_k(\mathbf{x})) + log(\pi_k) > log(f_l(\mathbf{x})) + log(\pi_l) \\ \iff & log\frac{f_k(\mathbf{x})}{f_l(\mathbf{x})} + log\frac{\pi_k}{\pi_l} > 0 \\ \iff & log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + \mathbf{x}^T\Sigma^{-1}(\mu_k - \mu_l) > 0. \end{aligned}$$

The classification rule is therefore linear in $\mathbf{x}$. This linear log-odds implies that the decision boundary between classes k and l is linear in $\mathbf{x}$, i.e. a hyperplane in p dimensions. It follows that the linear discriminant functions, given by

$$\delta_k(\mathbf{x}) = \mathbf{x}^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k,$$

Stellenbosch University https://scholar.sun.ac.za

are an equivalent description of the decision rule, with $C(\mathbf{x}) = \text{argmax}_k \delta_k(\mathbf{x})$. This metric arises when using the Mahalanobis distance metric to compute the distances to the class centroids (Tibshirani *et al.*, 2003).

Since the population class means, class proportions and the covariance structure are not available, these need to be estimated from the training data in order to make use of the above equation. The class proportions are estimated by $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the number of observations in class k. The class centroids are estimated by their sample means $\hat{\mu}_k = \frac{1}{N_k} \sum_{i \in C_k}^{N} \mathbf{x}_i$ and the covariances are estimated by the pooled sample covariance:

$$\hat{\Sigma} = \frac{1}{(N-K)} \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T.$$

When $p \gg N$, however, the matrix $\hat{\Sigma}$ is singular and therefore $\hat{\Sigma}^{-1}$ does not exist. One way to overcome this is to force the covariance matrix to be non-singular by implementing a specific form of Regularised LDA (Friedman, 1989). This form shrinks the pooled sample covariance matrix $\hat{\Sigma}$ towards the matrix formed by its diagonal elements. Therefore, $\hat{\Sigma}$ is replaced by,

$$\hat{\Sigma}(\lambda) = \lambda \hat{\Sigma} + (1 - \lambda) \text{diag}(\hat{\Sigma}),$$

where $\lambda \in [0, 1]$ controls the amount of regularisation that is applied.

When $\lambda = 0$, then Diagonal LDA is formed. In this, $\hat{\Sigma}$ is replaced by $\text{diag}(\hat{\Sigma})$, the inverse of which can be computed easily. By making the simplifying assumption that inputs are independent within each class, all the interaction terms can be ignored and therefore basic regularisation is also achieved. Although the assumption is not realistic, it does become useful in high-dimensional scenarios, since there is not enough data to estimate the dependencies between inputs.

The discriminant score for class k, under a diagonal covariance LDA model, is:

$$\delta_k(\mathbf{x}^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \bar{x}_{kj})^2}{s_j^2} + 2 \cdot log(\pi_k)$$

where $\mathbf{x}^* = (x_1^*, x_2^*, \cdots, x_p^*)^T$ is a test observation, s_j is the pooled within-class standard deviation of the j^{th} variable (assumed constant for all classes k) and $\bar{x}_{kj} = \frac{1}{N_k} \sum_{i \in C_k} x_{ij}$ is the mean of the N_k values of the j^{th} variable in class k. The centroid of class k is denoted by $\bar{\mathbf{x}}_k = (\bar{x}_{k1}, \bar{x}_{k2}, \cdots, \bar{x}_{kp})^T$ and therefore the first term in the above equation is the (negative) standardised squared distance of x^* to the k^{th} centroid. The second term adjusts the distance based

Stellenbosch University https://scholar.sun.ac.za

on the class prior probability, π_k, which can be estimated by the in-sample class-proportion, $\frac{N_k}{N}$, as above.

Let $\mathbf{x}^* = (x_1^*, x_2^*, \cdots, x_p^*)^T$ denote a test observation. The classification rule for this observation is then expressed as,

$$C(\mathbf{x}^*) = \text{argmax}_k \delta_k(\mathbf{x}^*).$$

Diagonal LDA will often outperform standard LDA in high-dimensional problems in terms of prediction; however, it uses all the variables and therefore is not ideal in terms of interpretation. Interpretability could be improved if only a small subset of the features were included in the model and therefore further regularisation is justified. Nearest Shrunken Centroids (NSC) is a closely related technique and is described below.

Nearest Shrunken Centroids

NSC stems from the nearest centroids classifier. The nearest centroids classifier will assign an observation to the nearest class mean (referred to as a centroid) (Tibshirani *et al.*, 2002). The mean for each of the K classes, with respect to every feature $j = 1, 2, \cdots, p$, is computed as follows:

$$\bar{x}_{kj} = \frac{1}{N_k} \sum_{i \in C_k} x_{ij}.$$

The centroid for class k is defined as the vector containing all p means, i.e. $\bar{\mathbf{x}}_k = (\bar{x}_{k1}, \bar{x}_{k2}, \cdots, \bar{x}_{kp})^T$.

The classification rule for a test observation $\mathbf{x}^*$ evaluated by a nearest centroid classifier is then expressed as:

$$C(\mathbf{x}^*) = \text{argmin}_l ||\bar{\mathbf{x}}_l - \mathbf{x}^*||,$$

which corresponds to the class mean with the smallest Euclidean distance to the new observation. After the appropriate standardisation, this classification rule is equivalent to the diagonal LDA classifier when classes are of equal size (Hastie *et al.*, 2009, p. 652).

NSC modifies the nearest centroid classifier further by the use of a shrinkage procedure which pulls all the class centroids in towards the global centroid (Tibshirani *et al.*, 2003). Those variables resulting in class-centroids whose distance to the overall mean is small, do not carry much discriminating power and can therefore be eliminated in favour of a sparser model. This shrinkage and selection is performed by a lasso-style regularisation.

Stellenbosch University https://scholar.sun.ac.za

Let d_{kj} denote a measure of distance between the class centroids and the global centroid, for each variable j, provided by

$$d_{kj} = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k(s_j + s_0)}, \tag{2.4}$$

where $\bar{x}_j$ is the overall mean (over all classes) for variable j, $m_k^2 = \frac{1}{N_k} + \frac{1}{N}$ and s_0 is a small positive constant, typically chosen to be the median of the s_j values.

The idea is to shrink d_{kj} toward zero using soft thresholding (which is similar to that of the lasso). The shrunken distance is given by

$$d'_{kj} = \text{sign}(d_{kj})(|d_{kj}| - \Delta)_+,$$

where Δ is the penalty parameter which can be found by cross-validation.

An alternative to the soft threshold is to use a hard-threshold leading to $d'_{kj} = d_{kj}I(|d_{kj}| \geq \Delta)$; however, Hastie *et al.* (2009, p. 653) state that a soft threshold typically performs better.

The shrunken versions of the centroids are then obtained by replacing the distance term on the left hand side of Equation (2.4) by the shrunken distance d'_{kj} and then rewriting the equation as follows:

$$\bar{x}'_{kj} = \bar{x}_j + m_k(s_j + s_0)d'_{kj}.$$

Notice that if $d'_{kj} = 0$, then $\bar{x}'_{kj} = \bar{x}_j$. If $d'_{kj} = 0$, $\forall k$, then $\bar{x}'_{kj} = \bar{x}_j$, $\forall k$ and therefore variable j does not play a role in the classification rule. For multiclass problems, the variables included in each discriminant score can be different for each class. Depending on the value of Δ, the number of variables that are essentially redundant may be large and therefore the dimensionality of the problem is reduced.

The usual diagonal LDA discriminant functions can be computed by using these shrunken centroids instead of the original centroids. The resulting NSC discriminant score is then given by:

$$\delta'_k(\mathbf{x}^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \bar{x}'_{kj})^2}{s_j^2} + 2 \cdot log(\pi_k).$$

The NSC classification rule is

$$C'(x^*) = \text{argmax}_k \delta'_k(x^*).$$

Nearest shrunken centroids is less affected by noisy variables than shrunken centroids, which improves the prediction accuracy. Another advantage is that

Stellenbosch University https://scholar.sun.ac.za

the reduced subset of variables that survived the soft-thresholding process will aid in interpretability in high-dimensional problems.

Hastie *et al.* (2009, p. 655) provides an example which effectively illustrates the dimension reduction capability of NCS when applied to gene expression microarray data. The example is briefly explained and illustrated in Figure 2.1.

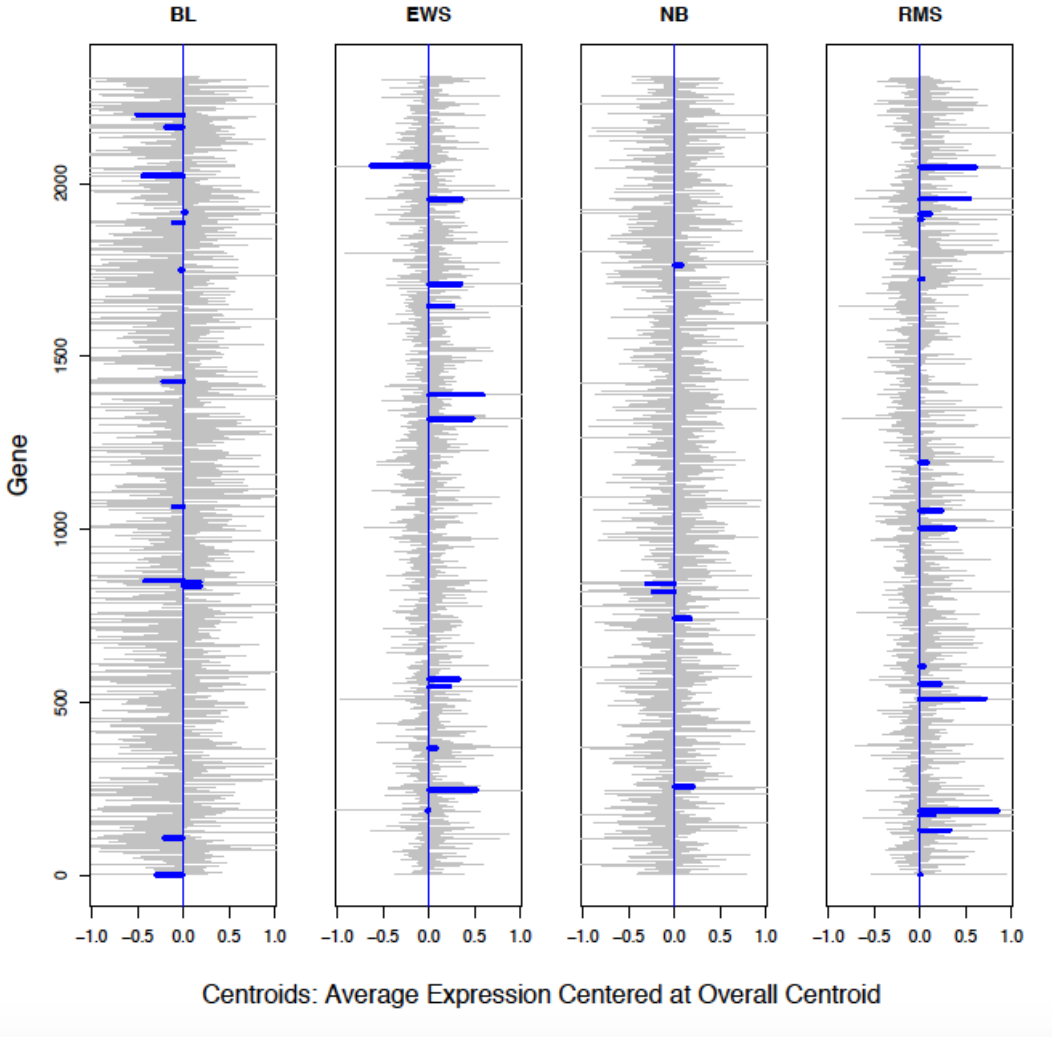

Figure 2.1: Four centroid profiles d_{kj} for the SRBCT (small, round blue-cell tumours) training dataset, which contains 63 observations and 2308 variables. The blue bars represent the genes that survive the thresholding. NSC was able to reduce the number of variables to 43 and still achieve zero error on the test set of 20 observations.

Stellenbosch University https://scholar.sun.ac.za

2.4 Feature Selection

Feature selection is another approach whereby one can reduce the dimensionality of a dataset. The difference between variable selection and feature selection is that variable selection reduces the number of original input variables to be used in fitting the model, whereas feature selection first transforms the original variables, to create new "features". This process is known as feature extraction. The number of features created r need not be equal to p. As a result of the transformation, r may be smaller than p, but even so, only a subset of the features may be required. Feature selection involves selecting the features that will be included in the subset used to fit a model.

The benefit of creating new features is that these new features can be more highly correlated to the response than any of the single input variables. In this regard it "summarises" the combined effect of many input variables, while constraining the dimensionality of the problem.

The drawback of feature selection is that the final model based on the features may be less interpretable than a model based on input variables.

Some feature selection methods are described below.

2.4.1 FastKNN

FastKNN is a feature extraction method that has been proposed and implemented by Pinto (2016), although the method is not documented in the literature. In high-dimensional cases, KNN can be computationally expensive because all the variables are taken into account in the distance calculation that is used for determining the nearest neighbours. FastKNN is an extension of KNN that is specifically adapted to high-dimensional problems. It can be viewed as a preprocessing step for KNN. The details of the algorithm are set out below.

Consider a multiclass classification problem where the training dataset consists of $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1, 2, \cdots, N\}$, where $y_i \in \{1, 2, \cdots, C\}$. Let $\mathcal{J}_c$ denote the set of indices of the observations that belong to class c, i.e. $\mathcal{J}_c = \{i \in 1, 2, \cdots, N : y_i = c\}$, for $c = 1, 2, \cdots, C$. Let $\mathbf{x}_0$ denote a test observation. The new features are constructed as follows:

- In each class, find the index $i_c \in \mathcal{J}_c$ of the training observation that is closest to the test observation. Denote these observations by $\mathbf{x}_{i_c}^{(1)}$.
- Compute the distance of the test case to the nearest cases in each class, i.e. $z_c^{(1)} = ||\mathbf{x}_0 - \mathbf{x}_{i_c}^{(1)}||$, $c = 1, 2, \cdots, C$.

Stellenbosch University https://scholar.sun.ac.za

- Repeat Steps 1 and 2, but now find $\mathbf{x}_{i_c}^{(2)}, j = 1, 2, \cdots, C$, the second closest neighbours of $\mathbf{x}_0$ in each class. Now compute the next set of features as the sum of the distances between $\mathbf{x}_0$ and $\mathbf{x}_c^{(1)}$ and $\mathbf{x}_c^{(2)}$, i.e. $z_c^{(2)} = z_c^{(1)} + ||\mathbf{x}_0 - \mathbf{x}_{i_c}^{(2)}||$.

- Repeat until the features $z_1^{(1)}, \cdots, z_C^{(1)}, z_1^{(2)}, \cdots, z_C^{(2)}, \cdots, z_1^{(K)}, \cdots, z_C^{(K)}$ have been computed in this manner. The final output is the transformed test case in the form of a vector consisting of $M = KC$ features.

The number of neighbours to find, K, should be chosen such that $M < p$ in order to achieve dimensionality reduction. In order to apply a KNN classifier to the test case, the observations in the training set also need to be transformed. This can be done by performing cross-validation. Firstly the training data are randomly partitioned into mutually exclusive folds. In the first iteration, every case in the first fold can be treated as if it were a test case and the above algorithm can be executed for every $\mathbf{x}_i$ in this fold. Repeating the procedure for all the folds yields a matrix $Z : N \times M$ of observations that have been mapped to a lower dimensional space. KNN can now be trained on Z rather than X, and the test case $\mathbf{x}_0$ classified based on this model. Notice that the computational burden of calculating distances in p dimensions is still incurred in the preprocessing step (both in transforming the test case, and the training set), however, the computational time of the KNN step is decreased.

In this method the features were formed by taking the response into account, therefore the feature extraction process is supervised. The technique in the next section transforms the features in an unsupervised manner.

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) was introduced by Karl Pearson more than a century ago (Pearson, 1901) and it remains one of the most popular dimension reduction techniques. It is an unsupervised method that is especially useful in data visualisation, since it is able to find a lower-dimensional configuration of the data that captures as much of the variation inherent in the data as possible.

Each principal component (PC) is a linear combination of the p original variables and each PC is orthogonal to every other PC. This can be interpreted as projecting the original variables onto a lower-dimensional subspace. The axes of the subspace are rotated in such a way that the features are in order of highest variance to lower variance. Because of this, it is often the case that much or most of the variation in the input space is explained by the first two PCs and the technique is therefore useful for data visualisation.

Stellenbosch University https://scholar.sun.ac.za

To find the coefficients of the linear combinations that form the PCs, the direction of maximum variation in the data must be determined. This can be done by performing an eigenvalue decomposition of the standardised sample covariance (or correlation) matrix, $S = \frac{1}{N-1}X^TX$. When the scaling factor $\frac{1}{N-1}$ is ignored, the decomposition takes the following form,

$$X^TX = V\Lambda V^T, \tag{2.5}$$

where V is a $p \times p$ matrix of orthogonal eigenvectors, such that $V^TV = I$. The matrix Λ is a $p \times p$ diagonal matrix of eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0$. From the above it follows that,

$$X^TXV = V\Lambda.$$

The matrix V is therefore the rotation matrix whereby the data are transformed to a new coordinate system. The vectors $\mathbf{v}_1, \mathbf{v}_2, \cdots \mathbf{v}_p$ (the columns of V) form the principal component directions. The eigenvalues are proportional to the variances of the principal components.

The above solution is equivalent to performing a singular value decomposition (SVD) on X and squaring the matrix of singular values. A description of SVD is given below, as well as the derivation of the aforementioned statement.

The SVD of a standardised matrix X can be written as

$$X = UDV^T,$$

where $U : N \times m$ and $V : m \times p$ are orthogonal matrices such that $U^TU = I$ and $V^TV = I$, and $m = \min(N-1, p)$ is the rank of X. D is an $m \times m$ diagonal matrix, containing the singular values $d_1 \geq d_2 \geq ... \geq d_m \geq 0$ as diagonal elements.

It follows that the matrix X^TX can be decomposed as follows:

$$\begin{aligned} X^TX &= VD^TU^TUDV^T \\ &= VD^2V^T. \end{aligned}$$

By comparing this to Equation 2.5, it is apparent that $\Lambda = D^2$ and that the rotation matrices are equal. Practically this means that for each principal component direction, the solutions obtained by the two different methods will only differ by a scaling factor equal to the variance of the corresponding PC.

The principal components of X (also referred to as principal component scores) are given by the columns of $Z = XV = UD$. The columns of V are referred to as the principal component loadings (or directions).

Stellenbosch University https://scholar.sun.ac.za

For data visualisation, the first two PCs can be plotted to obtain a 2-dimensional view of the data. An example of this is given by Figure 2.2. PCA is performed on a dataset of samples from 3 types of wine cultivars with 13 variables and the first two principal components are plotted (Forina, 1991). The plot shows that the cultivars can be separated fairly well by only using these two components.

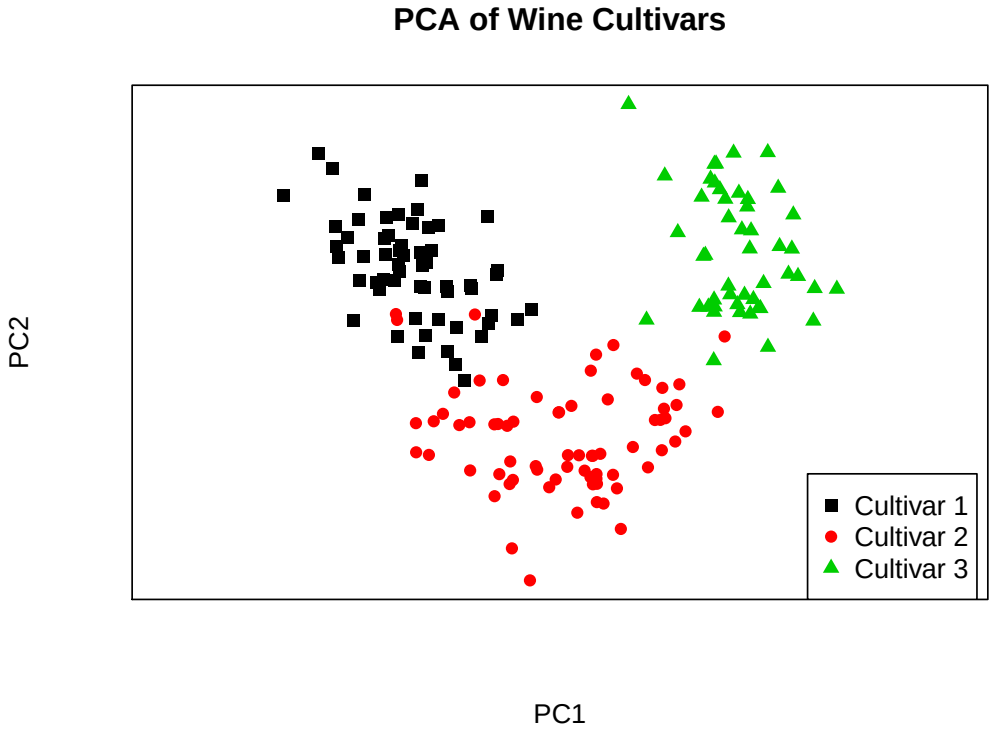

Figure 2.2: Plot of the first two principal components of the Wine dataset. The cultivars are separated adequately by these two dimensions.

Principal components can also be used in predictive models. Instead of basing the model on the original variables, a smaller number of the principal components can be used. This applies to both regression and classification scenarios. The number of principal components obtained from the eigenvalue decomposition is $m = \min(N-1, p)$, and therefore if $p > N$, then by using the PCs instead of the original variables, the dimensionality has already been reduced at this point. One could reduce the dimensionality further by discarding some of the columns of Z. It is common to discard those columns which explain the least variation, therefore only retaining the first few columns of the matrix Z. For exploratory data analysis, where there is no dependent variable, this is still the recommended procedure. In the presence of a response variable,

Stellenbosch University https://scholar.sun.ac.za

however, one can choose which PCs to retain in a different way and more will be said on this topic at a later point.

Before proceeding with the discussion of which PCs to choose, consider first the number of PCs to retain. This decision is somewhat subjective and there is no hard rule to be followed here. However, the eigen-decomposition has some additional useful by-products which may provide some guidance. The eigenvalues in the matrix Λ provide a measure to quantify the amount of variation explained by a principal direction. This can be expressed as a proportion of the total variation in the data.

The proportion of variation explained by the i^{th} principal component can therefore be defined as

$$\frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j} \text{ , for } i = 1, 2, \cdots, m.$$

It is also useful to look at the cumulative sum of these proportions to see the total amount of variation explained when including each additional principal component. In simple cases inspecting these proportions will be revealing enough to decide on the number of principal components to retain. For example, one might be fortunate enough to find that the variation explained by the first PC is 60% and the second is 30% and the remaining PCs make up the remaining 10%. Together the first two dimensions explain 90% of the variation, which might be deemed sufficient.

In most cases, however, the variation explained by successive principal components gradually dwindle down and it is not so easy to decide on a cut-off. To this end, the proportions can be plotted to create a scree plot. The rule of thumb with a scree plot is to find the "elbow" in the graph and then use one fewer than that number of principal components. Figure 2.3 illustrates the scree plot for the Wine dataset. The elbow seems to occur somewhere around component four, therefore only the first three components could be used in building a predictive model. The total variation explained by the first three components in this example is 66.5%

PCA has one major drawback when used for predictive purposes. Since it is an unsupervised method, it uses only the variance of the features to determine what the most "relevant" directions in the data are. When only the principal components with the highest variance are selected for further analysis, one implicitly makes the assumption that the response will vary most in the direction of the highest variation in the input space. In other words, it is assumed that the higher the variance of a given principal component, the more correlated it will be with the response. This is a strong assumption and there is no guarantee that this should be the case (Jolliffe, 1982, p. 302). Indeed, it is

Stellenbosch University https://scholar.sun.ac.za

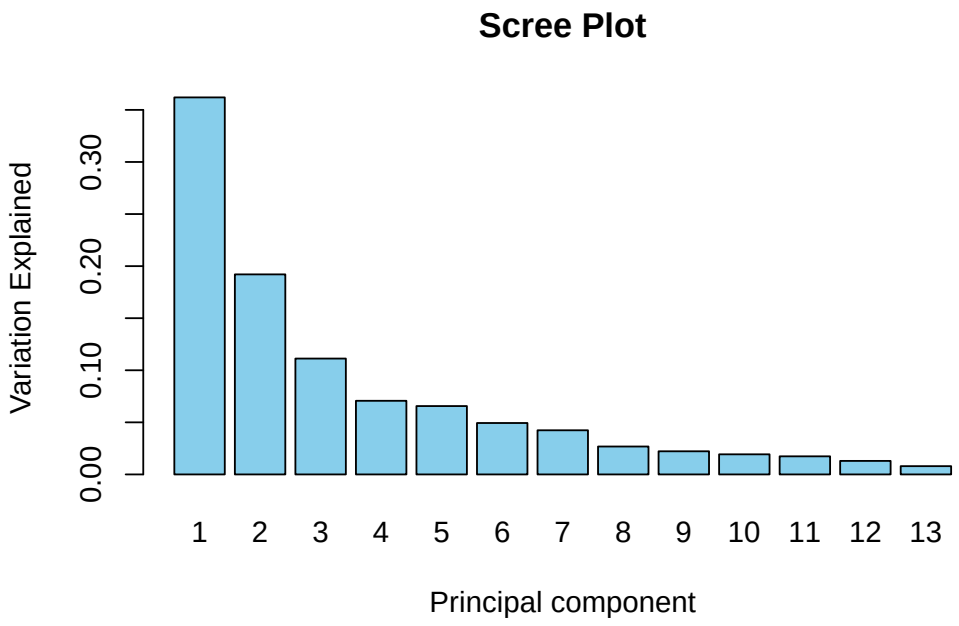

Figure 2.3: Scree plot of the variation explained by each principal component of the Wine data set.

possible that the principal component with the least variation might explain the variation in the response the best, as can be seen in Figure 2.4. Suppose a linear classifier is constructed based on only one principal component. For the data in the left panel of Figure 2.4, selecting the first principal component as the predictor will result in a perfect classification of the given sample. For the data in the right panel, however, the first PC would not be able to separate the two classes, even though it displays the highest variance. If the second PC is chosen instead, then the model would again be able to classify the sample correctly. Notice that the coordinates of the points are the same in the left panel and the right panel, only the classes of the points are different.

There is therefore some reason to believe that some improvement can be made by introducing supervision into the construction of the principal components. One method of doing so is to test all the principal components for significance once a model has been fitted and only retain the significant components (Jolliffe, 1982, p. 301). Another method would be to measure the correlation between the response and each principal component individually, and only choose the principal components with a moderate to strong correlation. Other methods that address this problem are Supervised PCA and Y-aware PCA, which are discussed below in more detail.

Stellenbosch University https://scholar.sun.ac.za

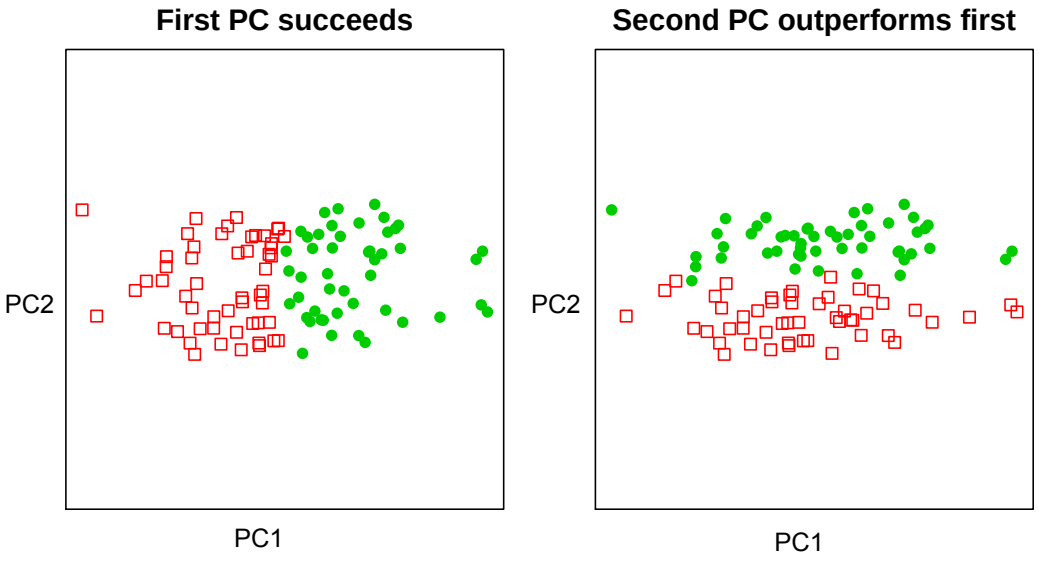

Figure 2.4: A simple binary classification example. On the left: the first principal component has the highest variance and is also the most correlated to the response. On the right: the first principal component has the highest variance, but the second is most correlated to the response.

2.4.3 Supervised Principal Component Analysis

Background

Consider a scenario where $\mathbf{y}$ is a vector of observed survival times of N patients, and $\mathbf{x}_1, \cdots, \mathbf{x}_p$ are the $N \times 1$ vectors of gene expression measurements given as input variables. In scenarios such as this it is typically the case that $p \gg N$. Suppose the aim is to find a subset of genes that can be used to predict the survival time of patients with a particular type of disease. Suppose it is believed that a particular disease is linked to certain (unmeasured) cell types which determine the average survival time of patients with that disease. For instance, in Figure 2.5 patients with Cell Type 1 live shorter on average than patients with Cell Type 2. Although the cell types are not directly measurable, the correlation amongst the gene expression measurements could be indicative of the cell type.

A model is required that is able to find the subset of genes that are correlated to the cell type, as well as deliver predictions on survival time. Although PCA is able to provide new features that exploit the correlation between the variables, not all the genes are related to the cell type affecting the survival time of patients with this particular disease. This means that genes exhibiting a high variance, irrespective of its correlation with survival time, will be included in the model. Due to its unsupervised nature, principal component regression (PCR) may therefore not perform well.

Stellenbosch University https://scholar.sun.ac.za

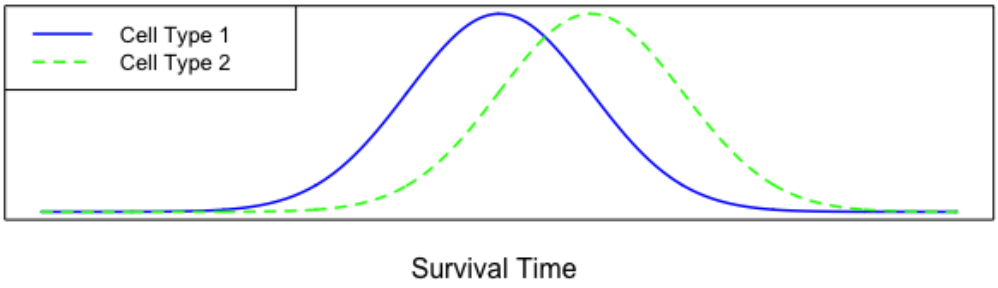

Figure 2.5: Underlying conceptual model. Source: Hastie *et al.* (2009, p.676).

Gene shaving is a method proposed by Hastie *et al.* (2000) aimed at reducing the set of genes to a small subset of highly correlated genes, whose average exhibit a high variance over the samples. The algorithm is related to the procedure followed by fastKNN, in the sense that the data are randomly partitioned into say r folds and each fold is considered in turn. In gene shaving, the first principal component is computed for the first fold and those genes with small loadings are "shaved" away from the input matrix before the next fold is considered. At each iteration, the first PC from the reduced set of genes (from the previous iteration) is computed and then additional genes are eliminated based on their loadings. The result is nested subsets of gene expressions. Each subset has stronger pairwise correlation and a higher variance of the largest principal component than its predecessor. Although this technique can eliminate genes from the dataset that contribute little to the principal component directions, it is unsupervised. As previously discussed, a high variation in the PCs does not necessarily imply a high correlation with the response. Some improvement can therefore be made by taking the response variable into account.

One such method is tree harvesting, which was introduced by Hastie *et al.* (2001). This consists of a hierarchical clustering component, as well as a predictive model. Features used in the predictive model are the average expression profiles for each cluster. It should be noted that here the genes are clustered, and not the observations. The clusters to be taken into account in the computation of the features are determined by forward stepwise selection. This method therefore clumps variables together in a supervised manner and the clusters that survive the selection process will be those that result in the greatest improvement in fit. PCA also averages the effect of multiple variables, but its shortcomings are that noisy variables are not excluded, and the first principal component may not be the feature with the strongest correlation with the response. Tree harvesting addresses both of these issues. Unlike PCA however, the combined effect of the variables is an equally weighted average, which does not account for the variation inherent in each variable. In addition,

Stellenbosch University https://scholar.sun.ac.za

the procedure requires much computation due to its hierarchical structure and the selection method.

The method discussed in the next section also addresses the limitations of PCA in a high-dimensional predictive scenario, but can be viewed as a simplified method of creating the clusters used in tree harvesting.

Supervised principal component analysis introduction

Supervised principal component analysis (SPCA) is an extension of PCA that introduces supervision to correct for some of the limitations of PCA. It was first introduced by Bair and Tibshirani (2004), however, related ideas are presented by Ghosh (2002) and Jiang *et al.* (2004). SPCA takes a semi-supervised approach wherein it uses variable screening in order to eliminate variables with a weak estimated correlation with the response, Y, before performing PCA on the reduced data set. The technique is semi-supervised since the response is only used in the initial screening process to determine the correlations, but thereafter the feature extraction is unsupervised. Bair *et al.* (2006) state that SPCA outperforms partial least squares (PLS) in survival prediction cases and attribute the success of the algorithm to the initial screening process of SPCA. By implementing the screening process, features with little correlation with the response (but potentially high variation) will be detected and excluded from the analysis. SPCA is used particularly when the number of predictors, p, is much larger than the number of observations, N, such as in gene expression mircoarrays. The random variable Y could be categorical, indicating the presence or absence of a particular medical condition in a patient, or it could be a continuous measure such as survival time (which may be censored).

Underlying model for SPCA

Latent variables are unobserved variables that are used to account for an interaction effect between correlated variables (Loehlin, 2004, p. 17). The assumption is made that the variables are correlated because they arise from common variables, $\mathbf{U}$, that are not directly measured (Murphy, 2012, p. 337).

Consider the case where Y is a continuous response. The motivation for SPCA can be cast into the framework of a latent variable model. Suppose the response is related to latent variables, denoted by the random vector $\mathbf{U} : t \times 1$, by a linear model,

$$Y = \beta_0 + \beta_1^T \mathbf{U} + \varepsilon,$$

where ε is an error term, independent of $\mathbf{U}$, and with mean zero. Additionally, the set of input variables can be split into two groups for $j = 1, 2, \cdots, p$ as

Stellenbosch University https://scholar.sun.ac.za

follows:

$$\begin{aligned} X_j &= \alpha_{0j} + \alpha_{1j}^T \mathbf{U} + \epsilon_j \text{ and } \alpha_{1j} \neq \mathbf{0} && \text{for } j \in \mathcal{P} \\ X_j &\text{ independent of } \mathbf{U}, && \text{for } j \notin \mathcal{P} \end{aligned}$$

where ϵ_j is an error term, independent of $\mathbf{U}$, and with mean zero.

The aim of a latent variable type of model is to obtain estimates of the sets $\mathcal{P}$, as well as the latent variables $\mathbf{U}$, so that a prediction model can be fit.

Probabilistically, this model can be expressed as follows for the variables in set $\mathcal{P}$:

$$\begin{aligned} \mathbf{U} &\sim N(\mathbf{0}, I) \\ \mathbf{X}|\mathbf{U} = \mathbf{u} &\sim N(\alpha_0 + W_{\mathbf{x}}^T \mathbf{u}, \sigma_{\mathbf{x}}^2 I) \\ Y|\mathbf{U} = \mathbf{u} &\sim N(\beta_0 + \beta_1^T \mathbf{u}, \sigma_y^2), \end{aligned}$$

where $\alpha_0 = [\alpha_{01}, \alpha_{02}, \cdots, \alpha_{0p}]^T$ and $W_{\mathbf{x}} : t \times p = [\alpha_{11}, \alpha_{12}, \cdots, \alpha_{1p}]$. The posterior probability can be expressed as

$$Y|\mathbf{X} = \mathbf{x} \sim N(\mathbf{x}^T \mathbf{w} \ , \ \sigma_y^2 + \beta_1^T C \beta_1),$$

where $\mathbf{w} = \Psi^{-1} W_{\mathbf{x}} C \beta_1$, $\Psi = \sigma_{\mathbf{x}}^2 I$ and $C^{-1} = I + W_{\mathbf{x}}^T \Psi^{-1} W_{\mathbf{x}}$ (Murphy, 2012, p. 405).

The remainder of the discussion focuses on the case where there is assumed to be only one latent variable.

Latent variable model example

A simple example of when a latent variable model might be used, is within the context of human resources. Suppose one would like to predict how long an employee will remain at a company (Y) and it is believed that the period of employment is linked to overall job satisfaction. Job satisfaction cannot be measured directly, however, and it is therefore a hidden variable (U). Instead a survey is distributed measuring a range of proxies that are presumed to be correlated with overall job satisfaction, for example, the employee is asked to indicate on a scale of 1 to 10 how likely they are to recommend their place of work to other job-seekers, or how happy they are with the coffee provided ($\mathbf{X}$). Presumably though, not all questions on the survey will be sufficiently correlated to overall job satisfaction, so one would like to discover the main contributors that lead to long-serving employees ($\mathcal{P}$), in order to accurately predict service time.

SPCA methodology

Stellenbosch University https://scholar.sun.ac.za

The first step in the SPCA algorithm is to estimate the set $\mathcal{P}$ by the use of sure independence screening (SIS). Since Y is assumed to be a function of U, if an input X_j is sufficiently correlated to the response, then it is assumed to form part of the set $\mathcal{P}$. The extent to which the variable must be correlated to the response in order to be included, will depend on a threshold value, θ, which can be determined through cross-validation. Therefore, for each input variable X_j, $j = 1, 2, \cdots, p$, the chosen score statistic, s_j, is calculated. Thereafter, a new data matrix X_θ is formed consisting only of the set $\hat{\mathcal{P}}$, i.e. the columns of X for which $|s_j| > \theta$. X_θ is of size $N \times r$ where $r < p$. The final step is to perform the usual PCA on this reduced data matrix, X_θ. The singular value decomposition yields

$$X_\theta = U_\theta D_\theta V_\theta^T,$$

where $U_\theta = (\mathbf{u}_{\theta,\mathbf{1}}, \mathbf{u}_{\theta,\mathbf{2}}, \ldots, \mathbf{u}_{\theta,\mathbf{m}})$ and $m = \min(N-1, r)$. The principal components are the columns of $Z_\theta = U_\theta D_\theta = X_\theta V_\theta$, where U_θ and V_θ are orthogonal matrices and D_θ is a diagonal matrix of singular values $d_1, d_2, \cdots, d_m$. Each principal component is an estimate of a latent variable. In the case of a single latent variable model, the first column of Z_θ will be the estimate $\hat{\mathbf{U}}$.

As in the case of ordinary PCA, the response corresponding to a new data case can be predicted by using the principal components in place of the original data in any chosen model. These features can be used in a regression or a classification model. Examples of these include linear regression and logistic regression respectively. The predictive model chosen may employ regularisation to the newly created features. Bair *et al.* (2006) suggest using only the first supervised principal component in the predictive model.

Methods related to SPCA

In SPCA, a hard threshold is applied to the variables in the screening step. Another idea is to consider a weighted form of SPCA wherein all the input variables are still included in the analysis, but some are down-weighted. The goal would be to decrease the influence of the variables that have a weak correlation to the response so that, when the PCA step is applied, these variables do not unjustly influence the principal directions. Y-aware PCA is one such method and is discussed in Section 2.4.4.

The above discussion focuses on the regression case, although SPCA can also be extended to the classification case. Here there is a question surrounding the method that can be used to measure the strength of the relationship between $Y \in \{1, 2, \cdots, C\}$ and $X_1, X_2, \cdots, X_p$. The SIS step of the algorithm was discussed for the binary classification case in Section 2.1.4. Under certain assumptions discussed therein, the squared univariate correlation can be used as the score function in the binary classification case. This is the approach taken in this thesis; however, a more general form of the score function is provided

Stellenbosch University https://scholar.sun.ac.za

in that section for when these assumptions are not met.

An interesting extension of the SPCA algorithm is iterative supervised principal components (ISPC). Although not pursued further in this thesis, a brief summary of the technique is provided here. ISPC has the benefit over SPCA of not needing to specify the value of the screening parameter (Piironen and Vehtari, 2018). It greedily finds the directions that best explain the variation in the response, by maximising the score function. At each iteration the algorithm finds the next best direction that is not already included in the model and is orthogonal to the other directions. The iteration stops when the next direction to be added is insignificant, as tested by a permutation test.

Favourable and unfavourable scenarios for SPCA

Although SPCA generally improves on the prediction performance of PCA, it does not do so in all cases. The effectiveness of the method will depend on the underlying data, as well as the rule used to determine whether a PC will be included in the final model or not (e.g., choosing the components with the highest variance, or choosing those most correlated to the response). To illustrate the scenarios in which SPCA will be beneficial, consider the top row of Figure 2.6.

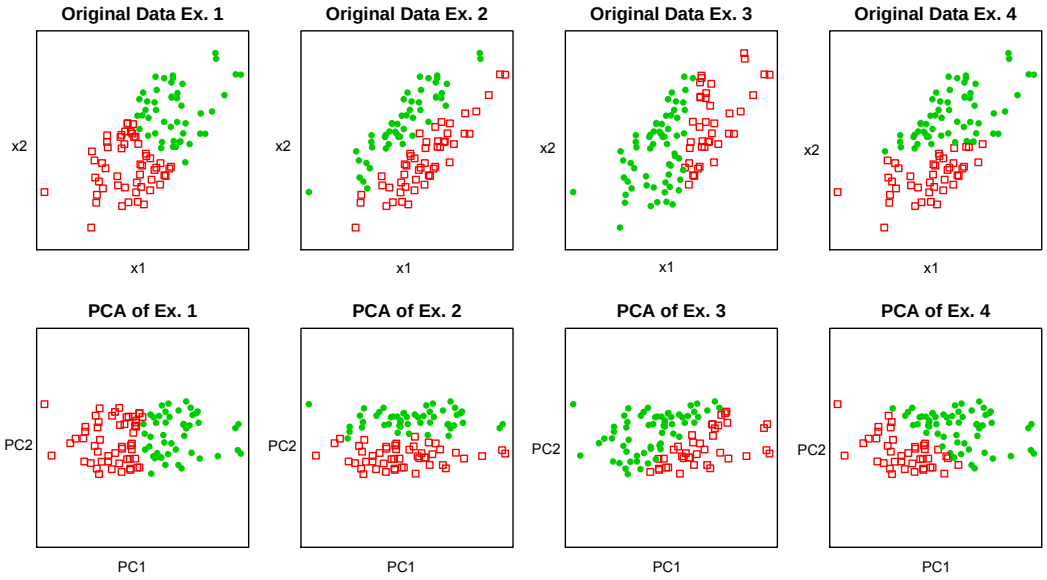

Figure 2.6: Different scenarios where SPCA either performs well or does not.

In this simplified example, the coordinates of the original data points in the graphs from left to right are kept fixed, but the classification of the points are varied. In all cases the two groups are perfectly separable by a linear classifier.

Stellenbosch University https://scholar.sun.ac.za

In the first two examples, both variables are needed in order to separate the untransformed data perfectly, whereas in Example 3 and 4, only one variable of the original data are needed.

For the sake of simplicity and ease of visualisation, suppose the goal is to reduce the dimensionality of the problem so that only one input/feature may be included in the final model. In the table below, 4 methods of feature selection are compared to each other. In each case the data were projected onto a new coordinate system and then only one principal direction was selected to build a basic linear classifier. The four techniques are based on PCA and SPCA. In the table below, "First PC" refers to projecting all the points onto the first principal direction (corresponding to the PC with the highest variance), and then classifying all the points according to a fitted linear decision boundary. "Single surviving SPC" refers to setting the threshold, θ, in SPCA to a value such that exactly one of the original variables survives the screening process. "Maximum correlated PC" refers to projecting all the points onto the principal direction corresponding to the PC with the highest correlation to the response. "CV max. correlated SPC" refers to determining θ via cross-validation and if more than one variable survive the screening process, then the one with the highest correlation to the response is used to build the classifier.

Table 2.1: Sample misclassification rates for each of the four examples in the previous figure.These examples try to illustrate the conditions when SPCA will be most useful and least useful.

PC Used	**Example 1**	**Example 2**	**Example 3**	**Example 4**
1. First PC	0.00*	0.46	0.21	0.20
2. Single surviving SPC	0.20	0.25	0.00*	0.00*
3. Max. correlated PC	0.00*	0.00*	0.21	0.20
4. CV max. correlated SPC	0.00*	0.00*	0.00*	0.00*

** Minimum misclassification rate per example.*

In Example 1, the decision boundary in the original input space will be a straight line with a negative gradient. Therefore, both X_1 and X_2 are needed to linearly separate the two groups. If PCA is performed, then each PC will be a linear combination of X_1 and X_2. Here, only the first principal component is needed to achieve 100% classification accuracy on the training set. In this example, the PC with the highest variance is also coincidentally the one maximally correlated to the response and therefore Model 3 performs equally well.

If one were to screen out one of the two variables in the first step of SPCA, then only X_1 or X_2 will be included in the model and the resulting classifier

Stellenbosch University https://scholar.sun.ac.za

will not be able to perfectly separate the two groups, as can be seen by the errors made by Model 2. When the threshold is instead determined by cross-validation, then both variables are retained. Model 4 performs just as well as Model 3 on the first example.

Consider now the second example. It is very similar to Example 1 in the sense that both original variables are needed to be able to construct a linear decision boundary to perfectly separate the data, which will make Model 2 less effective. In this example, the decision boundary will have a positive gradient. This is not of importance in itself, but note what happens when PCA is performed on the set of original data. Again, only one principal component will be required to separate the data, but this time, it needs to be the second principal component. Recall that it is commonly assumed that the first principal component will be the most correlated to the response. If the rule to select the final features to include in the model is based on this assumption, then the classifier based on the first principal component will perform dismally. This statement is supported by the high misclassification rate provided in Table 2.1. Model 3 correctly identifies the underlying structure of the data and therefore significantly improves on the performance of Models 1 and 2. Model 4 results in the same solution as Model 3 in this case and therefore it also performs well.

Consider now Examples 3 and 4. In the original data, the groups are perfectly separable based on X_1 and X_2 respectively. If PCA is performed here and only one PC is kept, then neither the first nor the second PC will perform as well as screening out the less correlated variable and using only the remaining variable in the model. This can be seen from the poor performance of Models 1 and 3, which do not implement any pre-screening. Model 2 and Model 4 result in the same solutions in these two cases, having eliminated X_2 in Example 3 and X_1 in Example 4, before projecting the data.

Although quite artificial, the above example effectively illustrates that the performance of some models are better on certain data structures than on others. Notice that Model 4 was able to obtain the minimum misclassification rate on all 4 examples automatically. By applying SPCA in this way, the problems that surfaced in the contrived example will be avoided and SPCA can consistently do at least as well as PCA.

2.4.4 Y-aware PCA

Another way of accounting for secondary principal directions that are more correlated with the response than the first is by performing Y-aware PCA. This technique can be described as *Response Scaled PCA*. It is similar to SPCA in the sense that it is a supervised approach to PCA. The difference is that SPCA discards variables in a discrete way, whereas Y-aware PCA shrinks the

Stellenbosch University https://scholar.sun.ac.za

effect of the less correlated variables. The variables are scaled in proportion to their univariate influence on the response and PCA is performed on the scaled set of variables. The motivation behind this choice is that variables with a smaller variation, but a significant influence on the response can be inflated so that when PCA is performed, the direction of most variation will be weighted toward this earmarked candidate. Instead of eliminating variables that have little effect on the response, one merely shrinks their influence.

To perform Y-aware PCA, the scaled set of variables is computed as follows:

$$\mathbf{x}_i^* = \beta_i(\mathbf{x_i} - \bar{x}_i\mathbf{1}) \text{ , where } \beta_i = \frac{\langle \mathbf{x_i}, \mathbf{y} \rangle}{\langle \mathbf{x_i}, \mathbf{x_i} \rangle}.$$

This has the effect of expressing X in terms of units of change in Y.

Ordinary PCA is then performed on the scaled matrix X^*. After finding the set of transformed variables, Zumel (2016) recommends significance pruning, by performing either an F-test, Chi-squared test or a permutation test, to select the number of principal components to retain. These pruned Y-aware principal components are of a reduced dimension, compared to the original problem and these may then be used in any selected predictive model.

The benefit of this method over SPCA is that there is no thresholding parameter which needs to be tuned. In addition, Y-aware PCA could lead to higher predictive accuracy compared to SPCA, when the principal components to be included in the predictive model are selected based on their variance, as opposed to their influence on the response (as is common practice). This is because in cases where the most influencing (supervised) principal component does not have the highest variance, Y-aware PCA will increase the variance of the directions with the greatest influence on the response, making them more likely to end up in the position of first or second principal component. These significant PCs are therefore more likely to be included in the final predictive model and are expected to improve the prediction accuracy of the model.

2.4.5 General Framework for PCA Type Methods

PCA, SPCA and Y-aware PCA can be thought of as special cases of a more general framework where ordinary PCA is performed on a weighted form of the original input matrix X.

Let $W : p \times p$ be a diagonal weight matrix which can be used to transform the set of variables X into a new set X^*, on which standard PCA can then be performed:

$$X^* = XW = \tilde{U}\tilde{D}\tilde{V}^T.$$

Stellenbosch University https://scholar.sun.ac.za

Note that if W is the identity matrix, then X^* will be equal to the original X matrix and the original PCA method is retrieved.

Supervised Principal Component Analysis
Suppose we form an indicator variable for each variable in X to indicate whether the score exceeds the threshold θ, $\psi_j = I(s_j > \theta), j = 1, \cdots, p$. Let $\psi_1, \cdots, \psi_p$ form the diagonal elements of the matrix W, with zeroes elsewhere. This will have the effect of eliminating all the variables that did not survive the initial screening. Performing PCA on the non-zero columns of the matrix X^* will result in the usual SPCA solution.

Y-aware PCA
Y-aware PCA does not exclude any variables, but rather scales them. To obtain the appropriate form of W, the univariate regression coefficients of each variable onto Y must be computed, and then W can be formed as the diagonal matrix of these coefficients, i.e.

$$W = \begin{bmatrix} \beta_{x_1} & 0 & 0 & \dots & 0 \\ 0 & \beta_{x_2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{x_n} \end{bmatrix}.$$

Multiplying X with W will result in a matrix whose first column is the first column of X, scaled by β_{x_1}, whose second column is the second of X, scaled by β_{x_2}, and so forth. Performing PCA on this matrix will yield the Y-aware PCA solution.

Viewing these methods as special cases of the more general framework, begs the question: What other weights are possible and more crucially, appropriate? Whereas SPCA employs a hard-threshold, it is perhaps possible to use a soft-threshold, such as a lasso type of shrinkage. One fairly ad-hoc idea is to start with the diagonal elements of W as the logit transformation of the score statistic:

$$w_j = \frac{1}{1 + e^{-|s_j|}} \text{ , for } j = 1, 2, \cdots, p,$$

where s_j is any score statistic that quantifies the strength of the relationship between X_j and Y, for example the correlation between these variables. Here $w_j \to \frac{1}{2}$ if $|s_j| \to 0$ and $w_j \to 1$ if $|s_j| \to \infty$. A more appealing measure would be to have $w_j \to 0$ if $|s_j| \to 0$, which motivates the construction of the modified weights:

$$w_j = \frac{2}{1 + e^{-|s_j|}} - 1 \text{ , for } j = 1, 2, \cdots, p.$$

Stellenbosch University https://scholar.sun.ac.za

A further possibility would be to re-express the w_j's relative to their maximum, i.e. to use

$$\tilde{w}_j = \frac{w_j}{\max(w_1, w_2, \cdots, w_p)}.$$

This will ensure that $\max_j\{\hat{w}_j\} = 1$ so that the most important variable receives a weight of 1.

A drawback of this class of methods however, is that building a model based on the newly created features will not yield interpretable results. One method that aims to achieve greater prediction accuracy by correcting for the noise in a dataset, while still maintaining interpretability, is preconditioning and this is the topic of the next section.

2.5 Preconditioning

When selecting a predictive model, for either regression or classification, the accuracy of the model is not always the only consideration to take into account. Depending on the purpose of the investigation, the interpretability of the model may be of a higher concern than the predictive accuracy. In high-dimensional settings, for example DNA microarray studies, interpretability is often of great importance, whereas interpretability may be of less importance in applications such as image classification.

In general, when the original input variables are transformed into features, the interpretability of the final model is lost. Therefore, although feature selection techniques, such as SPCA, may excel at reducing dimensionality and result in accurate models, the resulting models may be less interpretable.

Variable selection techniques on the other hand, usually result in interpretable solutions; however, in high-dimensional cases, they are typically computationally expensive and may not even be feasible when $p > N$, such as in the case of backward stepwise selection.

Techniques that combine shrinkage and selection, such as the lasso and NSC, require fewer computations and therefore have some success in higher dimensional cases. However, when $p \gg N$, their performance weakens as they struggle to distill the variables down to a subset of relevant variables. These problems are exacerbated when the response variable is noisy.

Ideally, one would like to find a predictor that is accurate, as well as a small subset of interpretable variables. While many of the methods previously discussed attempt to solve these two problems simultaneously, preconditioning is

Stellenbosch University https://scholar.sun.ac.za

introduced as a method which attempts to solve these problems separately by means of a two-stage approach (Paul *et al.*, 2008).

In the first stage, a consistent predictor of the response is found, $\hat{\mathbf{y}} = \hat{f}(X)$. At this stage, interpretability is not of importance and therefore any model, $f(\cdot)$, yielding high accuracy on the test set can be used. The output of this first step is the model estimates for the response in the training set, denoted by $\hat{y}$. The data pairs $(\mathbf{x_i}, \hat{y}_i)$, now represent de-noised versions of the original observations, and the data are said to be *preconditioned*.

In the second stage, an interpretable model is fit to the new data pairs $(\mathbf{x_i}, \hat{y}_i)$. The goal of this step is to perform variable selection in some way so as to reduce the dimensionality and promote interpretability. Having de-noised the response, the performance of variable selection methods such as the lasso, NSC and forward stagewise selection, may be improved. Indeed, Paul *et al.* (2008) show that under certain conditions, an appropriate subset of predictors can be identified when using a combination of SPC in stage 1 and the lasso in stage 2, whereas the lasso in isolation is unable to do so.

This technique has mainly been applied in the regression setting. However Paul *et al.* (2008) give a brief introduction to the classification case, wherein NCS is applied in stage 1 and forward stepwise classification in step 2. It is stated that although preconditioning does not necessarily improve the prediction accuracy for classification problems, the final model captures a greater proportion of the signal generating variables, compared to certain other selection methods.

2.6 Concluding Remarks

In this chapter, selected techniques relating to dimension reduction and regularisation were discussed. The topic of variable selection covered best subset regression, forward-and backward-stepwise regression, forward-stagewise regression, as well as sure independence screening. Ridge regression was discussed as a shrinkage method, while the lasso and NSC were discussed as techniques that combine shrinkage and selection. Reference was made to LDA, regularised LDA, diagonal LDA and shrunken centroids in this discussion. Under the topic of feature selection, PCA and SPCA were elaborated on, with specific attention being given to the way in which components are selected for inclusion into a predictive model. Reference was made to Y-aware PCA and a general framework for these methods was proposed. Finally, the preconditioning method was described. The growing need for interpretability accompanying the increasing dimensionality of datasets and complexity of models, calls for an investigation into methods that can provide accurate predictions, as well as

Stellenbosch University https://scholar.sun.ac.za

model interpretability. Preconditioning presents a simple, potential solution to this problem.

In the next chapter, preconditioning is compared to other selection and prediction techniques in a simulation study focussing on the classification problem. The behaviour of the preconditioning algorithm is explored when applied to data that is structured in ways that increase the difficulty of the classification problem.

Stellenbosch University https://scholar.sun.ac.za

Chapter 3

Simulation Study

This chapter describes a simulation study that was undertaken as part of the research. Various factors affecting the effectiveness of preconditioning are identified and further investigated by comparing the accuracy and interpretability of models when certain factors influencing the outcome are varied. The focus is on binary classification scenarios.

3.1 Introduction

The concept of preconditioning can cover a broad scope of two step algorithms - the first step yielding a preconditioned predicted response, and the second yielding an interpretable and accurate model. Paul *et al.* (2008) applied a preconditioning approach to a classification scenario by performing forward stepwise selection in a logistic regression model in the first step and in the second step, applied nearest shrunken centroids to the preconditioned training data to obtain a predictive model. Other combinations of techniques in step 1 and step 2 are however possible. It is of interest to see the impact of applying various techniques in the first and second stage of the algorithm, both in terms of accuracy and interpretability.

Furthermore, the overall concept of preconditioning may be more effective under certain conditions than under others. For example, does the technique perform better than a standard (non-preconditioning) technique when there is a greater degree of separability between the classes? How does the variance of the distribution from which input data are generated impact the performance of preconditioning?

A simulation study is therefore presented in this chapter in order to gain a better understanding of the usefulness of preconditioning under various circumstances. The term 'usefulness' here is used to allude to the dual-purpose nature of preconditioning, since it addresses the problems of finding an ac-

Stellenbosch University https://scholar.sun.ac.za

curate predictor and finding an interpretable predictor separately. Therefore, in order to perform a comparative study involving preconditioning, both the accuracy of predictions and the interpretability of the final model have to be taken into account. In the experiments included in this chapter, the error rate, the number of predictors included in the model and the number of signal variables among these (i.e., variables that truly influence the response) are used to compare variations of preconditioned algorithms, under varying conditions, to one another.

Since it is known from the outset that an interpretable model is one of the goals of preconditioning, models that transform the original variables in some way were not considered for the second stage model. Penalised Logistic Regression and Nearest Shrunken Centroids were chosen for the task. Other models with poor interpretability, such as Supervised Principal Components, were selected for the first stage model, and were also used, among others, in isolation to serve as a benchmark for model accuracy.

In order to test the models under varying conditions, data were simulated to control the degree with which the one group in the binary data differs from the second, and in what ways. More detail on the simulation of the datasets is provided in a later section.

The research questions which the study aims to address for a binary classification context, are listed below:

1. Based on accuracy and interpretability, how well does preconditioning compare to other standard methods?

2. How does the preconditioned model compare to others when the separation of the data are poor (i.e., the differences between the means of the two groups are small)?

3. How does the preconditioned model compare to others when the main distinction between the classes is a difference in the variances of the features?

4. Is there an interaction effect when the scenarios in the above points occur simultaneously? That is, is the effect on prediction accuracy compounded when the difference in class means is small and the class variation is large?

5. How is the performance and interpretability of the preconditioned model affected by increasing the signal-to-noise ratio (SNR) (compared to lasso)?

6. How is the performance and interpretability of the preconditioned model affected when it is trained on a larger training set?

Stellenbosch University https://scholar.sun.ac.za

7. Is there an interaction effect when the factors in the above points are varied simultaneously? That is, is the effect on prediction accuracy compounded when the SNR is low and the training set size is small?

8. How is the performance of the precondition-based model impacted when using different predictive models in the first step of the preconditioning algorithm (i.e., by changing the model used to obtain the preconditioned response)?

9. How is the performance of the precondition-based model impacted when using different predictive models in the second step of the preconditioning algorithm (i.e., by changing the model that is trained on the preconditioned response and results in the final model form)?

3.2 Experimental Design

In order to answer the above questions, an experiment was designed which is able to test some of the effects of the changing factors in combination so that the influence of the experimental design on the results of the study can be minimised. The process is illustrated by Figure 3.1 and described in detail in the subsections to follow.

3.2.1 Approach to Answering Research Questions

To address each of the nine research questions listed in the above section, the following processes were followed:

1. Preconditioning is compared to standard methods, namely penalised logistic regression, supervised principal components and nearest shrunken centroids, in each of the scenarios described below

2. The fitted models are applied to datasets where the mean vector of one of the classes differs from that of the other class.

3. The models are applied to datasets where the covariance matrix of one of the classes differs from that of the other class.

4. The models are applied to datasets where the two groups differ with respect to both their means and their covariance matrices.

5. The models are applied to various datasets where the number of signal variables are altered relative to the total number of variables. (The ratio of these two factors is called the signal-to-noise ratio (SNR).)

6. The models are applied to various datasets where the number of training observations is varied.

Stellenbosch University https://scholar.sun.ac.za

7. The models are applied to datasets where both the SNR and the number of observations are varied.

8. Over all scenarios, the impact of using different models in the first step of preconditioning is tested by using KNN, LR, SPC, LDA or NSC in step 1.

9. Over all scenarios, the impact of using different models in the second step of preconditioning is tested by using PLR or NCS in step 2.

Stellenbosch University https://scholar.sun.ac.za

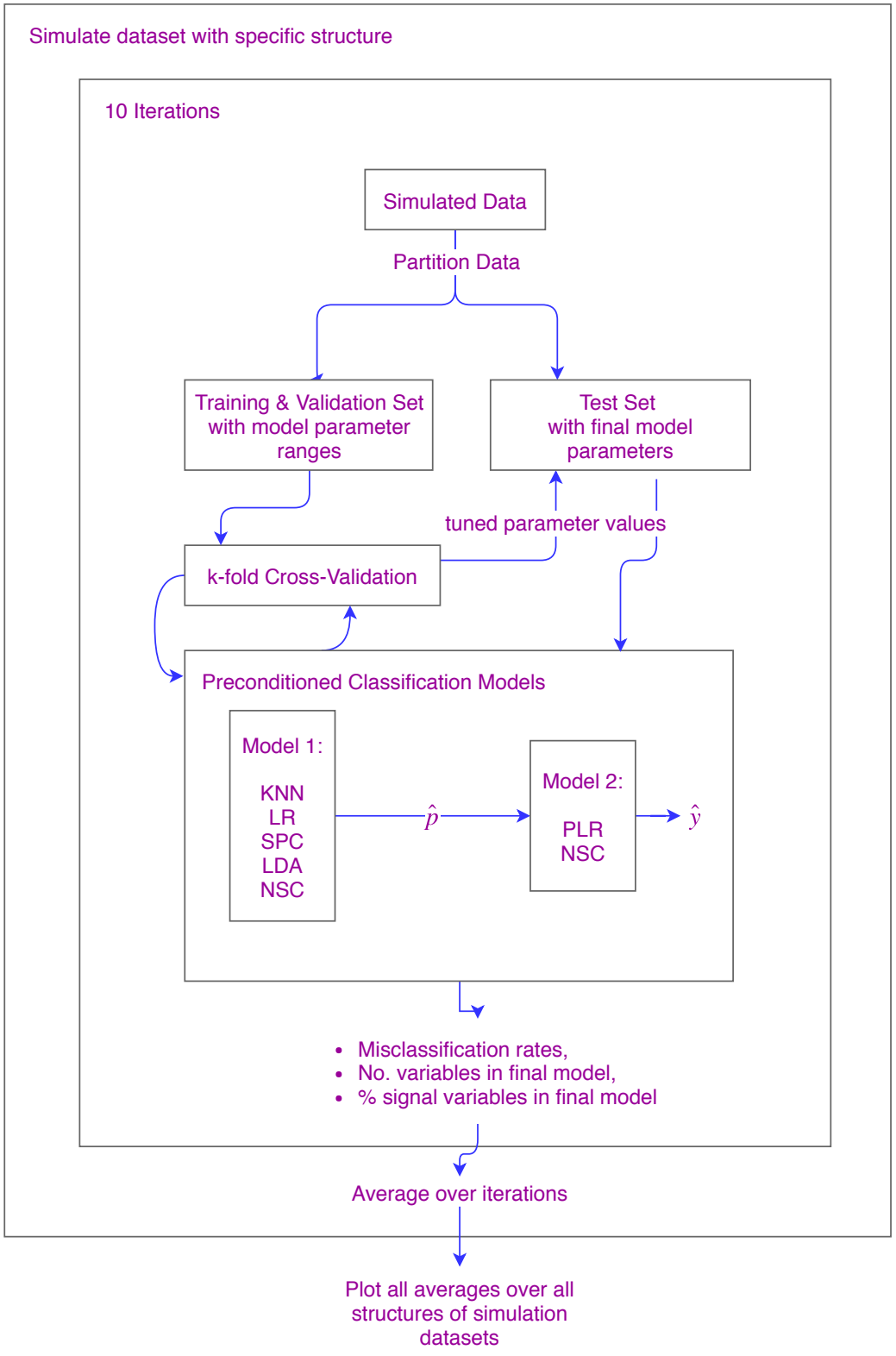

Figure 3.1: Simulation study setup.

Stellenbosch University https://scholar.sun.ac.za

The simulation study was performed by using the R programming language (R Core Team, 2017). Some models used were available as packages; however, where this was not the case, custom functions were created to perform the task. The main functions necessary to perform the study include a function to perform preconditioning, one to cross-validate global parameters, one to simulate different sets of data and one to compare preconditioning techniques to other standard classification techniques. These functions are briefly described below.

3.2.2 Preconditioning Function

Packages to implement preconditioning are available for the regression case, but no equivalent is available for a classification context and therefore had to be created. The method described by Paul *et al.* (2008) was used as a starting point; this can be found in Appendix B. In this function, a parameter can be set to choose the combination of models in step one and step two of the algorithm. The options for the first model, which yields the initial estimates of class probabilities, are: KNN, LR, SPC, LDA and NSC. The options for the second model, which delivers the final predictions and model coefficients, are: penalised logistic regression (PLR) and NSC. Any model parameters required are also passed in as function arguments.

If the second model is chosen to be penalised logistic regression, then the predicted class probabilities computed on the data from the first model are converted to the range $(-\infty, \infty)$ via the logit transformation: $y = log(\frac{\pi}{1-\pi})$. These are used, together with the original input variables, to train a PLR model. The model estimates of the response are then converted back to unit range, $[0, 1]$, via the inverse logit transformation: $\pi = \frac{e^y}{1+e^y}$. The final classification is made by the rule:

$$Y = \begin{cases} 0, & \text{for } 0 \leq \hat{\pi} < 0.5 \\ 1, & \text{for } 0.5 \leq \hat{\pi} \leq 1 \end{cases}.$$

If nearest shrunken centroids is selected as the second model, then the predicted probabilities obtained from step one are converted to binary classifications by the rule above. These binary values, together with the original input variables, are used to train an NSC model. The fitted probability estimates are then also converted to classifications.

For both of these methods, the number of misclassifications based on the validation set, the variables included in the final model, as well as the number of non-zero model coefficients are returned by the R function.

Stellenbosch University https://scholar.sun.ac.za

This function does not perform any tuning of global parameters. This is done by an overarching cross-validation function described below.

3.2.3 Cross-Validation

The global parameters required by the above function need to be optimised. Hastie *et al.* (2009) state that cross-validation (CV) must be performed over the entire sequence of steps when tuning parameters; therefore the function described above was applied repeatedly to varying training and validation sets and over a grid of parameters.

In R, many functions have the built-in capability to cross-validate its own parameters; however, cross-validation at every step is a greedy approach and the optimal global parameters may not be found. Another argument in favour of a single over-arching cross-validation is that the validation data will only be used once, as opposed to being used at every step in the sequence of modelling steps (as would be the case if the built-in cross-validation functionality is used). In the latter case, the model sees the validation data prematurely and we may get an overly optimistic view of the training error. An overly optimistic training error could result in the wrong parameter values being singled out and a less generalisable model being fit.

Cross-validation is therefore performed as follows:

- For each model parameter to be tuned, a likely possible range of values is selected beforehand.
- A list is made of every combination of all the values in the pre-selected ranges of likely parameter values.
- The training data are partitioned into 5 folds. The algorithm iterates over the 5 folds, keeping one fold separate for validation purposes and using the remaining 4 as training data.
- For each of the 5 folds, the algorithm iterates over every set of parameter values. The previously mentioned preconditioning function is trained on 4 of the folds, using the current set of parameter values. The misclassification rate on the validation fold is computed and recorded. For each set of parameters, the average misclassification error over the 5 folds is calculated and used as an estimate of the model's performance for that set of parameters.
- The final set of model parameters is selected as the set which minimises the average misclassification error over all the validation folds.

Stellenbosch University https://scholar.sun.ac.za

A note on the minimisation of misclassification error: due to the discrete nature of misclassification rates, it is possible that more than one set of parameter values could minimise the average validation misclassification rate. Where more than one set was found, the first in the range was selected.

3.2.4 Simulated Data

The function for generating the input data produces a matrix of random values from a normal distribution. This is achieved by using the *rnorm()* function in R, which simulates random values from a normal distribution, with mean and variance as stated in the arguments of the function. The matrix contains observations from two groups and includes both signal variables as well as noise variables. Figure 3.2 illustrates symbolically what this matrix would look like. The first r columns are the columns that make a distinction between Group 1 and Group 2, i.e. the signal variables. The remaining $p - r$ columns have no direct relationship with the response variable and are therefore considered to be noise. Suppose Group 1 contains N_1 observations and Group 2 contains N_2 observations. Let $N = N_1 + N_2$ be the total number of observations.

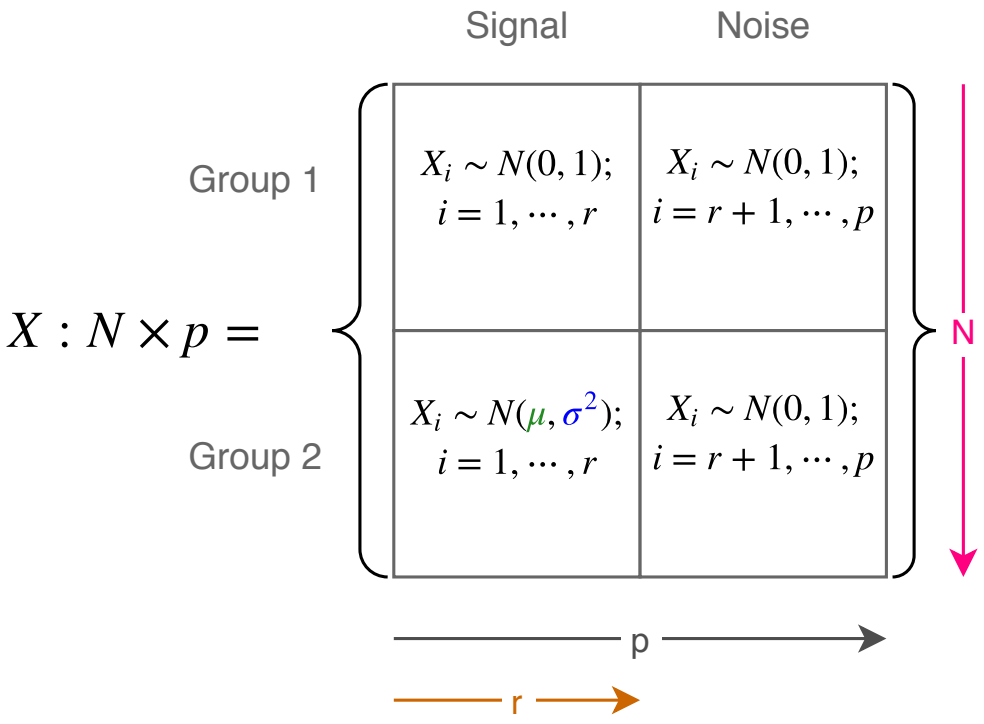

Figure 3.2: Data simulation setup.

Stellenbosch University https://scholar.sun.ac.za

Now, define $\mathbf{X}$ as a random matrix with elements, X_{ij}, distributed as follows:

$$X_{ij} \sim \begin{cases} N(0,1) & \text{for } i = 1, \cdots, N_1 \; ; \; j = 1, \cdots, r \\ N(\mu, \sigma^2) & \text{for } i = N_1 + 1, \cdots, N_2 \; ; \; j = 1, \cdots, r \\ N(0,1) & \text{for } i = 1, \cdots, N \; ; \; j = r+1, \cdots, p \end{cases}.$$

The only distinction between the two groups therefore is in the mean and variance with which the first r columns of Group 2 are simulated. In the experiments presented in this chapter, equal numbers of observations for Group 1 and Group 2 are produced, i.e. we take $N_1 = N_2$.

Many of the research questions can be addressed by altering the parameters with respect to which the data are simulated and evaluating the model(s) on the various datasets. By this simulation set-up, research questions number 2, 3 and 4 can be addressed by changing the values of μ, σ^2. Research questions 5, 6 and 7 can be addressed by changing the values of r, N. Questions 1 and 8 can be addressed by changing the models which are fitted to the data.

The R function written to simulate the datasets for given parameters N, p, r, μ and σ^2 is named *Gen.Data()* and is provided in Appendix B.

The first simulation study presented compares the results obtained by various models when evaluated on the simulated datasets where μ, σ^2 are varied. The values chosen for the mean of Group 2 are $\mu \in \{0.1, 0.25, 0.5, 1\}$ and the values of the variance are $\sigma^2 \in \{0.1, 0.5, 1, 1.5, 2\}$. The combined total number of training and test observations is $N = 250$. The hold-out test set is 25% of the data and therefore the split between training and test is 187/63 (where $N_1 = 93 \approx N_2 = 94$) and the number of features and signal variables are $p = 1000$ and $r = 50$ respectively.

Figure 3.3 illustrates an example of how the underlying data varies for the first study. It also gives structure to the layout of the grid of results to be presented in the first study: the mean of Group 2 increasing from left to right and the variance of Group 2 decreasing from top to bottom.

The second simulation study compares the results obtained by various models when evaluated on the simulated datasets where the number of training observations, N, and the number of signal variables, r, are varied. The values chosen for the total number of observations are $N \in \{50, 100, 250, 500, 750, 1000, 2500\}$, with the test set again forming 25% of each of the sets. The values chosen for the number of signal variables are $r \in \{50, 100, 250, 500\}$. The total number of variables is fixed at $p = 1000$ and the distribution of the signal variables of Group 2 is $N(0.5, 1)$.

Stellenbosch University https://scholar.sun.ac.za

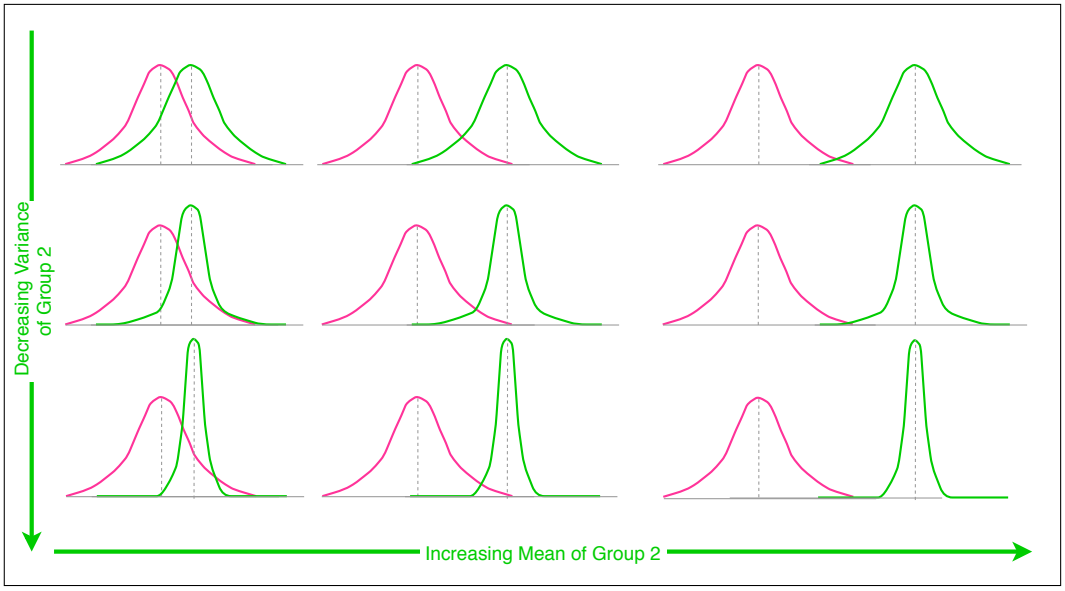

Figure 3.3: Study 1: In this study the distribution of the signal variables in the second group is varied, both in terms of mean and in terms of variance. The distribution of the first group remains fixed.

Figure 3.4 illustrates an example of how the underlying data matrix varies for the second study. Each rectangle represents a matrix of data that grows longer as N increases from left to right. The red part of the rectangle represents the proportion of noise variables in the data, which decreases from top to bottom. This representation also gives structure to the layout of the grid of results to be presented in the second study.

Randomness is a key part of of the effectiveness of using simulated data; however, to avoid an over-reliance on any particular dataset, the whole study on all of the variations of datasets, including the cross-validation step, was repeated 10 times and the mean and variance of each measure computed. Where variation is high within the 10 iterations, it can be said that the combination of models used is unduly influenced by the random variation in the data and therefore is less successful at capturing the underlying structure of the data than a technique with a comparable mean, but a lower variance.

3.2.5 Other Models Used for Comparison

In order to create a benchmark for the performance of preconditioned models, 3 other models were applied to the same datasets over the same iterations. The models selected for this task were: penalised logistic regression (PLR), nearest shrunken centroids (NSC) and logistic regression applied to supervised principal components (SPC). In this latter technique, only the SPC with the highest correlation to the response was used as a feature in a logistic regression

Stellenbosch University https://scholar.sun.ac.za

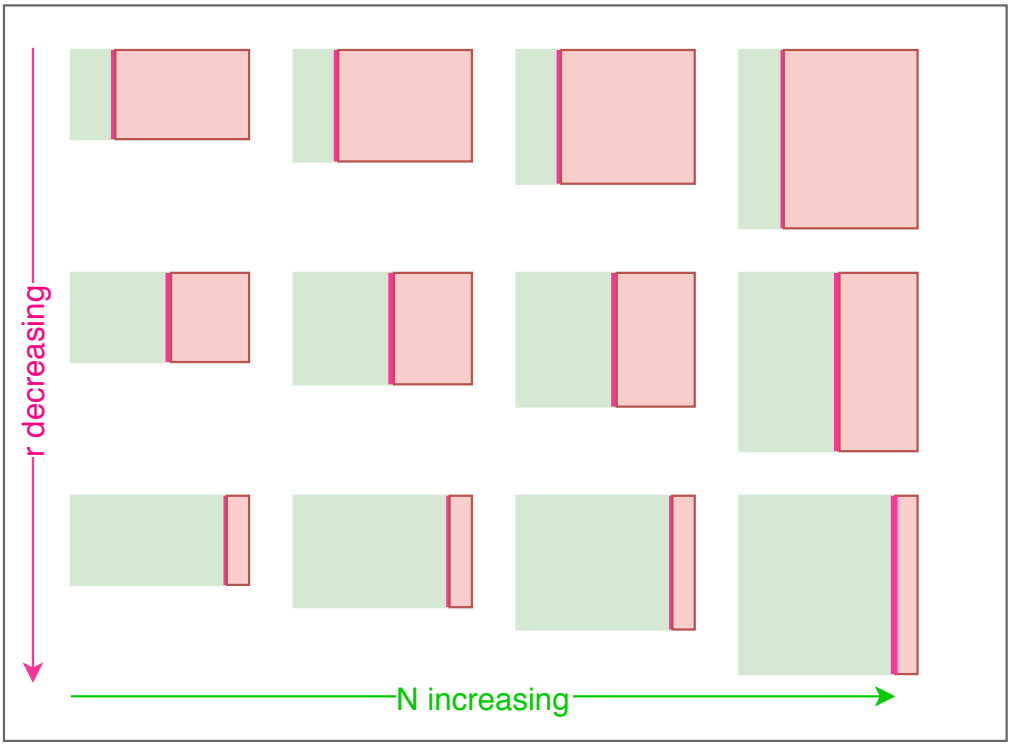

Figure 3.4: Study 2: In this study the number of training observations is increased relative to the (fixed) number of variables. In addition, the number of signal variables is increased relative to the fixed total number of variables.

model. Let $h = \text{argmax}_j(\text{corr}(\mathbf{z}_{\theta,j}, \mathbf{y})$, where $\mathbf{z}_{\theta,j}$ is the j^{th} column of the matrix Z_θ obtained from SPC. Then although the explicit form of this model is $log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 z_{ih}$, it can also be written in terms of the underlying input variables as

$$\begin{aligned} log\left(\frac{p}{1-p}\right) &= \beta_0 + \beta_1 \mathbf{x}_{\theta,i}^T \mathbf{v}_h \\ &= \beta_0 + \beta_1 \sum_{j=1}^{r} x_{\theta,ij} v_{jh} \\ &= \beta_0 + (\beta_1 v_{1h}) x_{\theta,i1} + (\beta_1 v_{2h}) x_{\theta,i2} + \cdots + (\beta_1 v_{rh}) x_{\theta,ir}, \end{aligned}$$

where $\mathbf{v}_h$ is the column vector of the matrix V_θ that is obtained when performing SVD on the reduced matrix X_θ. The right hand side of this equation is a linear function of $\mathbf{x}_{\theta,i}$, but unlike the usual logistic regression case, only two parameters need to be estimated, namely the intercept term β_0 and the common factor β_1, as the vector $\mathbf{v}_h$ is known from the computation of the SVD. Therefore, when using only one SPC in a predictive model, an interpretable solution is possible, although it should be noted that the r original input vari-

Stellenbosch University https://scholar.sun.ac.za

ables that survived the SIS step of the algorithm, will be present in this form of the final solution. Ideally, r should be small to aid in interpretability.

The performances of the above-mentioned models are provided alongside those of the preconditioned models. This set of models is referred to as 'standard models' or 'non-preconditioned models' in the sections to follow.

3.3 Results

3.3.1 Study 1: Varying the Relative Distribution of Group 2

In this study the mean and variance of Group 2 in the simulated dataset were varied, as described in the previous section. The aim of this simulation study is to explore the way preconditioning reacts to such changes in order to discover whether there are scenarios to which it is more well suited, or where it has the advantage over other more standard techniques. The following 3 figures depict the average misclassification error rates of the test set, the average number of variables included in the final model, and the average proportion of signal variables included in the final model over 10 iterations respectively. Paul *et al.* (2008) refer to this as the proportion of 'good predictors'. Suppose the subset of input variables known to be signal variables (by the simulation design) is denoted by $\mathcal{P}$ and the subset selected by the model is denoted by $\hat{\mathcal{P}}$. Let the number of variables used in the construction of the final model be denoted by $p_{\hat{\mathcal{P}}} = \sum_{j=1}^{p} I(X_j \in \hat{\mathcal{P}})$. The proportion of 'good predictors' can then be expressed as

$$\begin{aligned}\mathcal{G} &= \frac{\sum_{j=1}^{p} I(X_j \in (\mathcal{P} \cap \hat{\mathcal{P}}))}{\sum_{j=1}^{p} I(X_j \in \hat{\mathcal{P}})} \\ &= \frac{1}{p_{\hat{\mathcal{P}}}} \sum_{j=1}^{p} I(X_j \in (\mathcal{P} \cap \hat{\mathcal{P}})).\end{aligned}$$

In each of the 10 iterations, the test set consisted of the ceiling of 25% of the observed cases that were simulated. Since $N = 250$, this amounted to 63 cases.

Stellenbosch University https://scholar.sun.ac.za

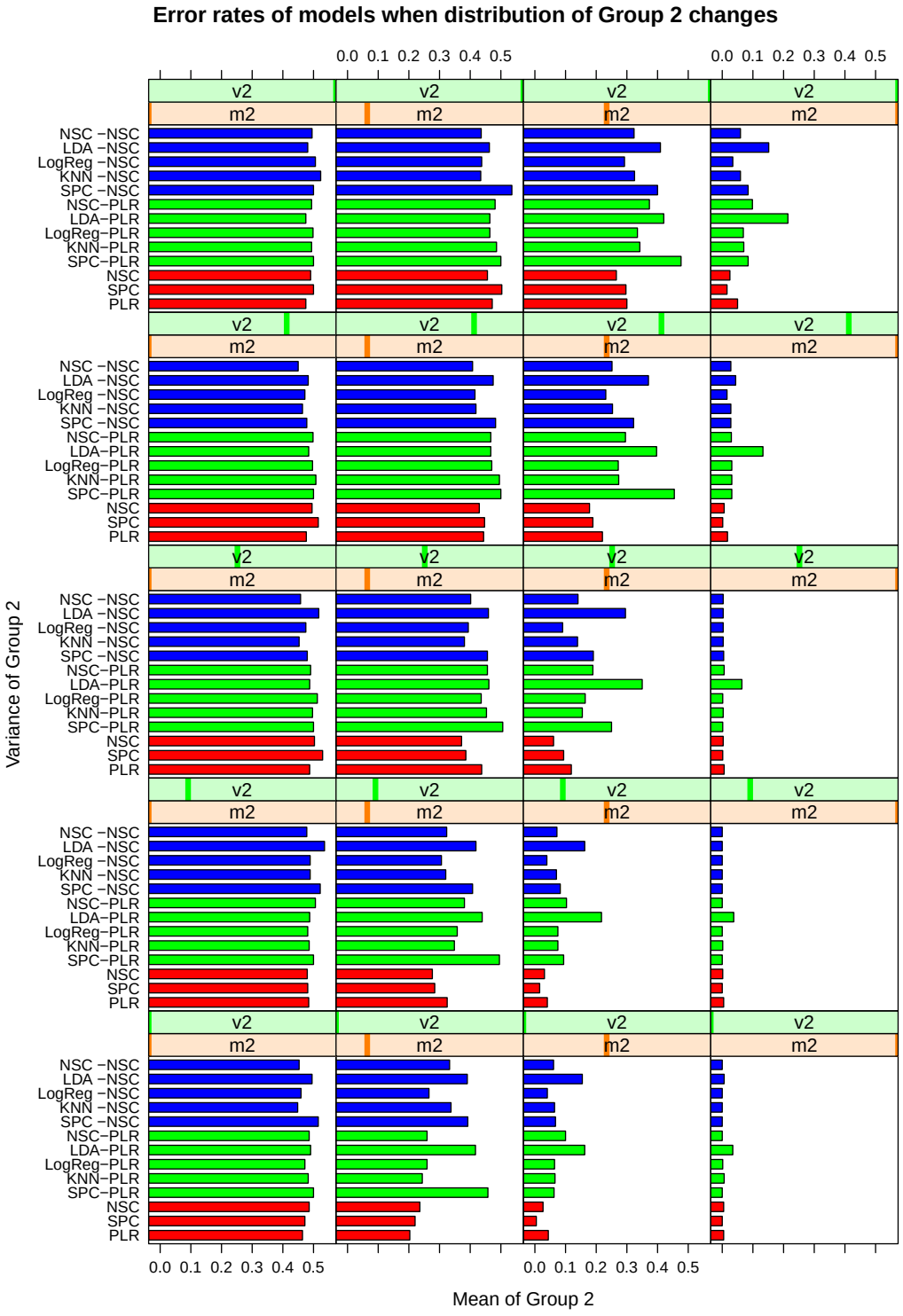

Figure 3.5: The graph above shows the average misclassification rates on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 (m2) increases from left to right, while the variance of Group 2 (v2) decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Average misclassification rates

From the first column of Figure 3.5 it can be seen that all models perform poorly, with misclassification rates close to 0.5, when the mean of the distribution of Group 2 is close to that of Group 1. Decreasing the variance of the distribution of Group 2 does little to aid classification when the means are nearly equal, although some minor improvements can be seen in certain techniques, for example NSC-NSC and KNN-NSC.

Moving to the next column of the figure, it can now be seen that with a small increase in the mean of Group 2, there is an improvement in the results. The improvement increases as the variance of Group 2 becomes smaller (moving from top to bottom in column 2). It seems that certain models are better than others at distinguishing between the groups in the low-variance, small-mean case. Specifically, when looking at the green group, the misclassification rates of NSC-PLR, LogReg-PLR and KNN-PLR were around 20% lower than their companions LDA-PLR and SPC-PLR.

When looking at the third column of the graph, the difference in performance between the various models becomes clearer. LDA-NSC, LDA-PLR and SPC-PLR consistently have the highest misclassification rates, whereas LogReg-NSC, NSC, SPC and PLR consistently have the lowest.

From the fourth column of the graph it is apparent that a greater difference in mean between the two groups improves the performance of all the models significantly, especially when the variance of Group 2 is also low. LDA-PLR stands out as the model with the highest misclassification rate for multiple scenarios.

In general, the group of non-preconditioned models (in red) perform better than the preconditioned models. The difference in performance between preconditioned and non-preconditioned models grows as the mean of Group 2 increases. This indicates that for more difficult classification problems, where the groups are not well separated, preconditioned models have a comparable performance to that of non-preconditioned models.

Furthermore it can be noted that when the variance of Group 2 is high (towards the top of the graph), the group of models using nearest shrunken centroids as the second model (in blue), perform slightly better than the preconditioned models which use penalised logistic regression as the second model (in green). As the variance drops, this effect seems to be reversed.

The variances of the misclassification rates are presented and discussed in Appendix A, Figure A.1. To briefly summarise the results contained therein: this

Stellenbosch University https://scholar.sun.ac.za

plot illustrates that the bias-variance trade-off argument does not extend to classification, as discussed in Chapter 1. This is because when bias is towards the wrong side of the decision boundary (i.e. misclassification rates are high), then an increase in variance can lead to improved results. It can also be seen that where fair accuracy is possible (column three), the variance bars follow the same pattern as the average error bars, implying that poor performance in this area is coupled with variable results.

Lower misclassification rates are desirable, but may come at the cost of including a greater number of variables in the model, causing a decrease in interpretability. This aspect is illustrated and discussed next.

Average number of variables in final model ($p_{\hat{\mathcal{P}}}$)
Please note that in Figure 3.6, $p_{\hat{\mathcal{P}}}$ refers to the number of original input variables that were selected and used in the final model. In particular, for SPC this means that it is the number of variables that survived the screening process, not the number of features (supervised PCs) that were used in the predictive model (only the single most correlated supervised PC was used). It should also be noted that an NSC model retains different sets of variables for each class in the (binary) classification problem. The number $p_{\hat{\mathcal{P}}}$ in this case refers to the total of these two sets and this applies to all preconditioned models with NSC in the second stage. For PLR and the preconditioned models with PLR in the second stage, this is the number of predictors with non-zero (penalised) coefficients.

From Figure 3.6 it is clear that using nearest shrunken centroids as the second model in the preconditioning algorithm (in blue) leads to a more complex solution, as more variables are included in these types of models. The preconditioned models using penalised logistic regression as the second model, as well as the non-preconditioned models, achieved quite drastic dimension reduction results in nearly all cases (NSC being the only exception).

Generally it is interesting to note that the difference in structure between the two groups in the data had little impact on the average number of variables included in each model. With minor exceptions, the pattern of the bars looks fairly similar for all the different data scenarios.

Greater variance in Group 2 prompted greater dimension reduction in LDA-NSC and LogReg-NSC, but lower dimension reduction in NSC.

Some models were more influenced by the increase in the mean of Group 2 than others: it prompted greater dimension reduction in NSC and, to a lesser extent, in LogReg-PLR. The opposite is true for LogReg-NSC, where the dimension reduction decreased with an increase in mean.

Stellenbosch University https://scholar.sun.ac.za

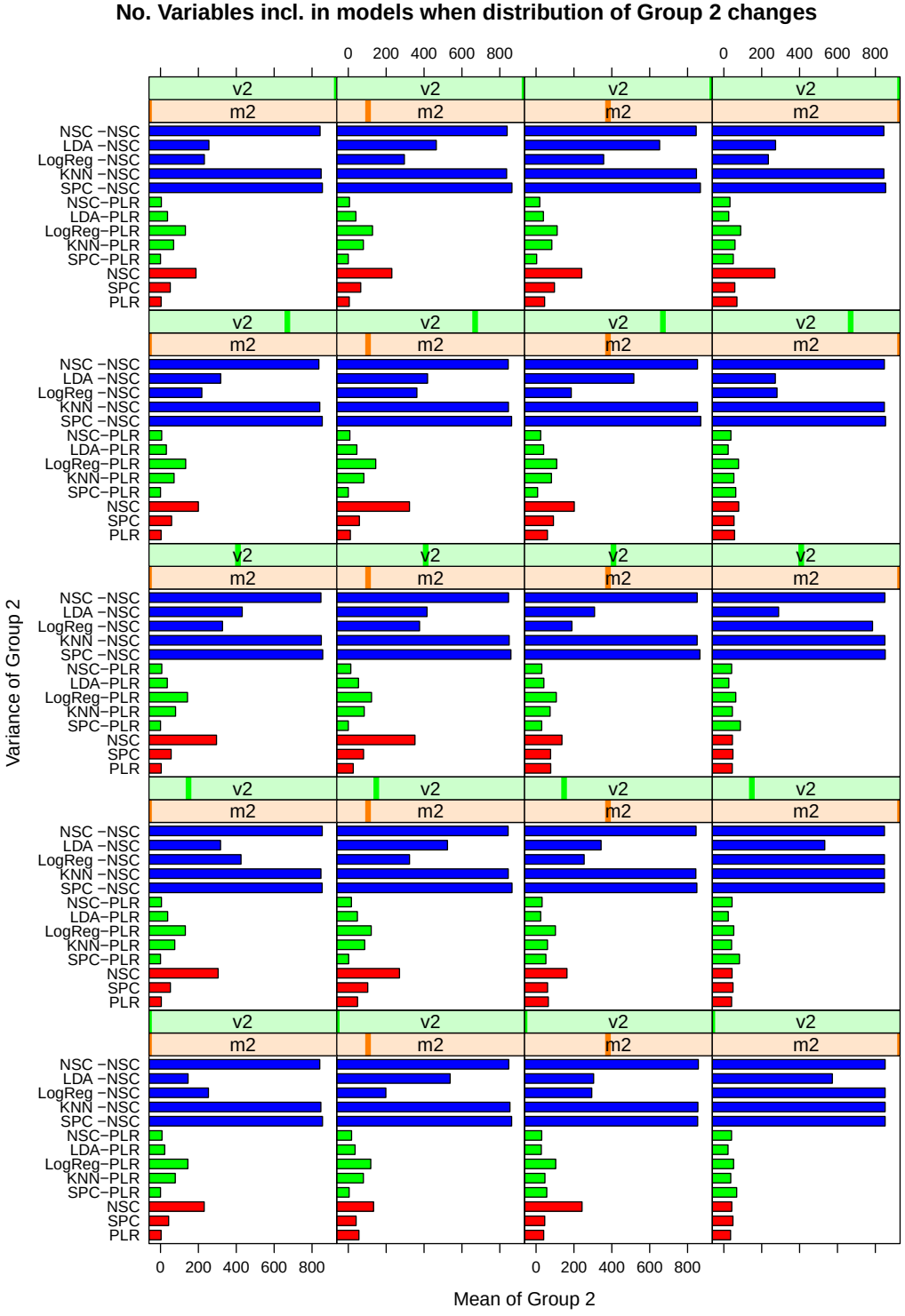

Figure 3.6: The graph above shows the average number of variables included in the final model $p_{\hat{\mathcal{P}}}$ (excluding the intercept term). The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 increases from left to right, while the variance decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

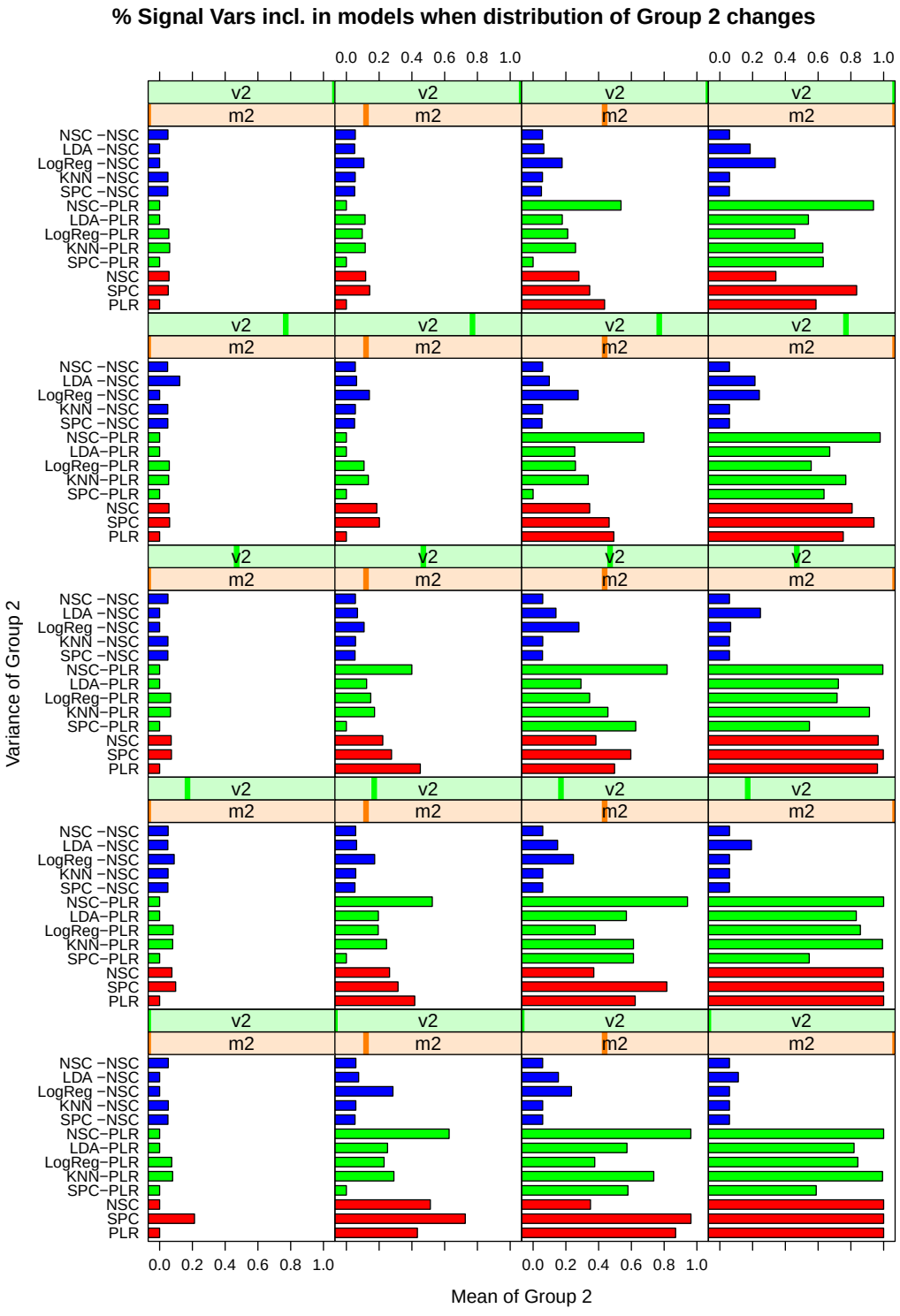

Figure 3.7: The graph above shows the average of the proportion $\mathcal{G}$. The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 increases from left to right, while the variance decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Overall, PLR, SPC-PLR and NCS-PLR resulted in the greatest dimension reduction.

Taking the error rates into consideration while viewing the graph, additional insights can be made. The non-preconditioned models were able to achieve good accuracy while simultaneously achieving significant dimension reduction results. Among the preconditioned models, using PLR as the second model achieved comparable dimension reduction to that of the non-preconditioned models, although error rates were slightly higher. The most notable preconditioned models based on error rates and dimension reduction were NSC-PLR, LogReg-PLR and KNN-PLR.

The variances of the number of variables included in the final model are presented and discussed in Appendix A, Figure A.2. The main conclusion to be made from these results is that the preconditioned models that achieved the greatest average dimension reduction (NSC-PLR, LogReg-PLR and KNN-PLR) also maintained a low variance thereof. LDA-NSC, LogReg-NSC and NSC had extremely high variances, although the average dimension reduction was fair.

In most cases the significant increase in dimensionality when using NSC as the second model, as opposed to PLR, overshadows the slight improvement of error rates. It should be noted that although NSC, LogReg-NSC and LDA-NSC had some success with simultaneous accuracy and dimension reduction in some scenarios, the number of variables included in the model showed very high variance over the 10 iterations (see Appendix A, Figure A.2).

Average proportion of 'good predictors' ($\mathcal{G}$)

In Figure 3.7, the proportion of signal variables included in the final model can be seen to increase as the groups become more clearly separable, with a greater difference in mean and a lower variance (towards the bottom right-hand corner of the graph). It seems that a difference in mean has a greater impact on the proportion of signal variables captured than the value of the variance.

The low proportion of signal variables included in the preconditioned models using NSC in the second phase (in blue), can be attributed to the large number of total variables included in the final model.

The proportion of signal variables captured by the non-preconditioned models and the preconditioned models that use PLR in the second phase seem comparable.

The models that ranked among the highest in terms of proportion of signal

Stellenbosch University https://scholar.sun.ac.za

captured include: NSC-PLR, SPC, PLR and KNN-PLR.

The variances of $\mathcal{G}$ are presented and discussed in Appendix A, Figure A.3. The main conclusion to be made from these results is that the variance is linked to the size of $p_{\hat{\mathcal{P}}}$ and also depends on whether the average $\mathcal{G}$ is close to zero (low variance), close to one (low variance) or in a mid-level range (high variance). It does however appear that for a given average value of $\mathcal{G}$, the green set of models provides a lower variance, with NSC-PLR, KNN-PLR and PLR performing particularly well in terms of the average and the variance of $\mathcal{G}$.

Conclusion

In this section, the main overall results of Simulation Study 1 are summarised.

NSC had a relatively low error rate for many of the scenarios; however, $p_{\hat{\mathcal{P}}}$ was often nearly double or more the size of some of the other models. Considering the variables are chosen per class, it is possible that the number of variables within a particular class will be comparable to that of other techniques (especially when the classes are poorly separated). The main concern is interpretability, and if that goal is achieved, the actual number of predictors is of less importance. For NSC, the average $\mathcal{G}$ was higher than those of many other models in cases where classes were poorly separated, but lower than those of many other models where classes were well separated.

NSC-NSC, LDA-NSC, KNN-NSC and SPC-NSC achieved low to moderate error rates for many of the scenarios; however, $p_{\hat{\mathcal{P}}}$ was among the highest in all the scenarios. Consequently the proportion $\mathcal{G}$ was low in all scenarios. These preconditioned-NSC models are therefore not recommended as suitable classifiers. The remaining technique in the blue group of models, LogReg-NSC, had somewhat more success: it achieved some of the lowest error rates, and the measures $p_{\hat{\mathcal{P}}}$ and $\mathcal{G}$ were comparable to that of NSC in some, but not all of the scenarios. NSC performed marginally better with respect to all three of these measures though, and in addition, requires less computation. It therefore seems that NSC is not suitable as a model for use on preconditioned data. It seems as though NSC can however be applied as a preconditioner. The model NSC-PLR achieved fair error rates and a low $p_{\hat{\mathcal{P}}}$ in most scenarios. It also attained the highest $\mathcal{G}$ for many of the scenarios, but not those where the groups were poorly separated. Therefore, NSC-PLR is a good model to apply when the group distributions differ somewhat in terms of mean and/or variance. The models LogReg-PLR and KNN-PLR exhibited many of the same characteristics mentioned for NSC-PLR; however, their performances in terms of these 3 measures were worse.

It is interesting to observe that the error rate of SPC-PLR decreases drastically from the third column to the fourth. This is accompanied by a small

Stellenbosch University https://scholar.sun.ac.za

increase in $p_{\hat{\mathcal{P}}}$, but a dramatic increase in $\mathcal{G}$. Therefore, irrespective of the variance of the distribution of Group 2, a sizeable difference in the means of Group 1 and Group 2 leads to a significant improvement in the overall performance of this model. In cases where the groups are poorly separated, this model does not perform well.

SPC and PLR are two models with similar behaviour in terms of the 3 performance measures used. Both models produced error rates and $p_{\hat{\mathcal{P}}}$ towards the lower end of the spectrum. In addition, they were able to produce proportions of 'good predictors' that were among the highest in nearly all the scenarios. In SPC only the single most correlated PC was included in the predictive model and therefore interpretation will also be possible. Therefore these models are comparable to the rather impressive performance of NSC-PLR: their error rates are slightly lower, but their proportions of good predictors are also slightly lower. The stability of the results seem fairly similar for the first two measures, but the variance of $\mathcal{G}$ is mainly higher for SPC than it is for PLR or NSC-PLR.

It seems that using PLR in the second stage of preconditioning (preconditioned-PLR) yields models that perform reasonably well to well, with the exception of LDA-PLR. This model consistently obtained some of the highest error rates, despite the fact that $p_{\hat{\mathcal{P}}}$ was relatively low and $\mathcal{G}$ was reasonably high.

Taking all three of these measures into account, it appears that the models providing the best balance include, in no particular order, NSC-PLR, SPC and PLR.

3.3.2 Study 2: Varying the Number of Observations and Signal to Noise Ratio

In this study, the distribution of Group 2 was kept fixed, but the number of observations in the dataset, as well as the signal-to-noise ratio of the dataset were altered. The same models as before were applied to these simulated datasets.

The following 3 figures depict the average misclassification error rates on the test set, the average number of variables included in the final model and the average of the measure $\mathcal{G}$ described in the previous section over 10 iterations respectively. In each of the 10 iterations, the test set consisted of the integer part of 25% of the observations in the simulated data. Since $N \in \{50, 100, 250, 500, 750, 1000, 2500\}$, the test set sizes used were $\{12, 25, 62, 125, 187, 250, 625\}$. The results are discussed below.

Average misclassification rates

Stellenbosch University https://scholar.sun.ac.za

In Figure 3.8 the average misclassification rates increase as the signal-to-noise ratio decreases (from top to bottom). It can also be seen that the rates decrease as the number of observations in the dataset increases (from left to right). All models, with the exception of LDA-PLR, perform well when both the number of observations and the signal-to-noise ratio are high (top right-hand corner). Conversely, the performance of all models is worse when both are low (bottom left-hand corner).

It can also be observed that the preconditioned models that use NSC in the second phase (in blue) have significantly lower misclassification rates than those that use PLR in the second phase (in green) when N is small. As N decreases, the error rates of the green set of models fall in line with those of the other two sets of models.

NSC and SPC have error rates similar to those of the preconditioned models that use NSC in the second phase, while PLR performs in line with the preconditioned models that use PLR in the second phase.

LDA-NSC and LDA-PLR consistently have high error rates when the number of observations is very low.

NSC-PLR is the only model that was unable to identify the correct signal variables when the signal-to-noise ratio was very low but the number of variables was high.

Arguably, the models with the best performances were KNN-NSC, LogReg-NSC, NSC and SPC.

The variances of the misclassification error rates are presented and discussed in Appendix A, Figure A.4. To briefly summarise the findings regarding the variances: no model displayed high variance consistently. The variance seems to be roughly coupled with the size of the average error rate.

Again, misclassification rates only are insufficient to measure the performance of preconditioned models. The following two graphs address the interpretability part of the problem.

Stellenbosch University https://scholar.sun.ac.za

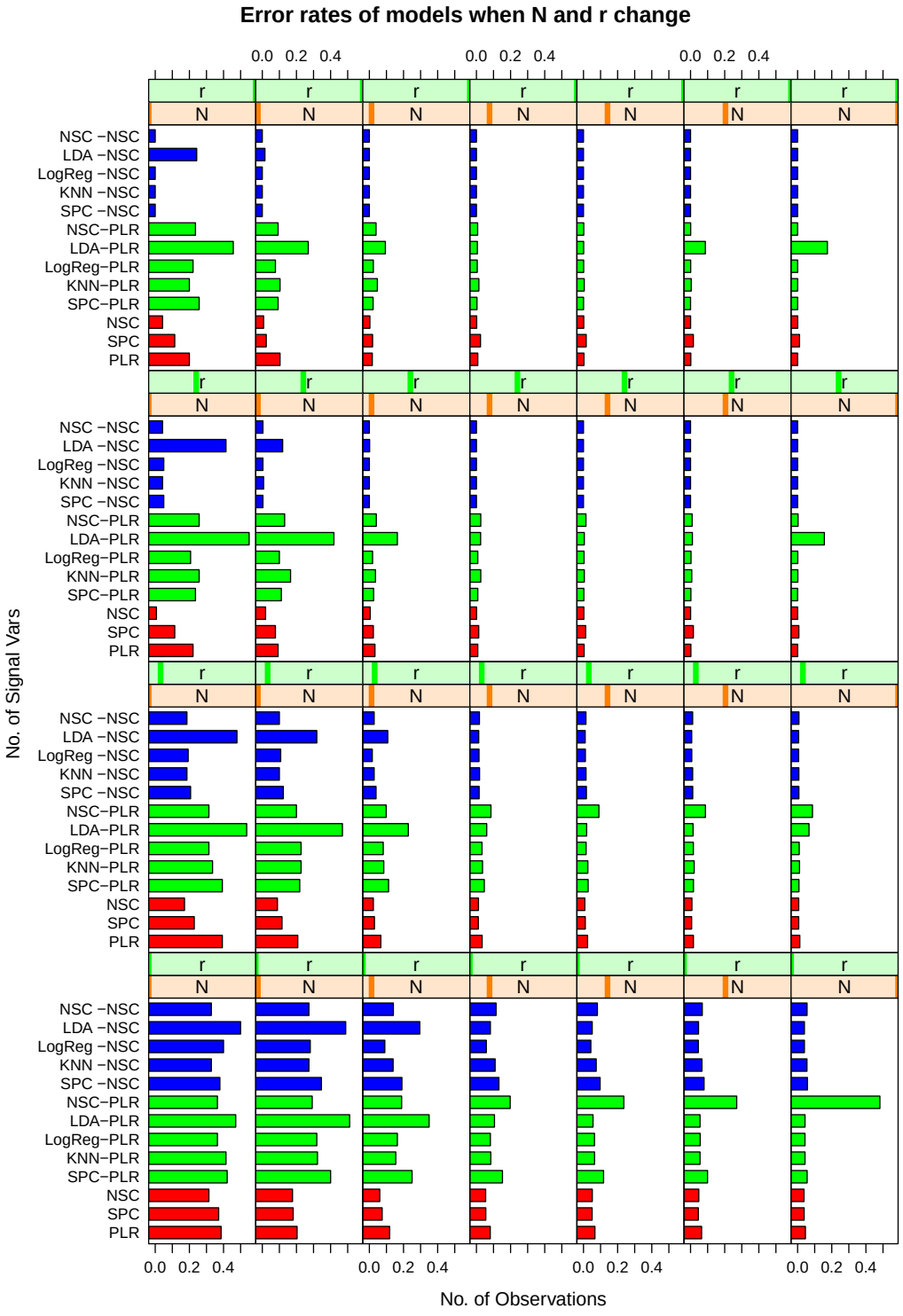

Figure 3.8: The graph above shows the average misclassification rates of 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of training observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

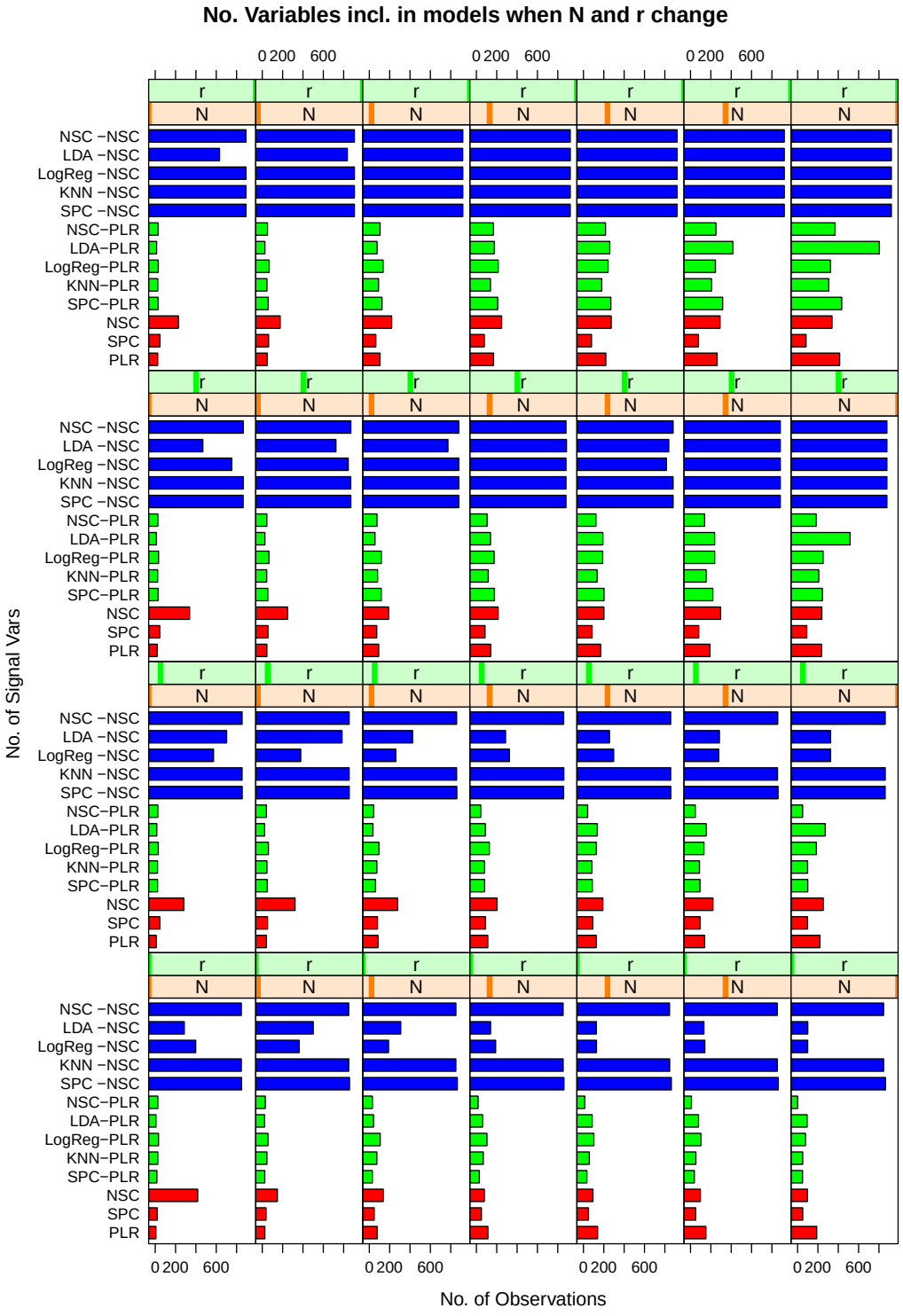

Figure 3.9: The graph above shows the average number of variables included in the final model $p_{\hat{\mathcal{P}}}$ (excluding the intercept term). The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Average number of variables in final model ($p_{\hat{\mathcal{P}}}$)
In Figure 3.9, it can be seen that as the number of training observations increases, so does the number of variables in the model. This behaviour is contrary to what is expected, since one would expect that more training cases would result in a model that is more certain of which variables give rise to the signal in the data. However, when the data are noisy, such as in this case, a model with more training cases might have more examples of noisy cases and therefore include more variables to compensate for its uncertainty in identifying the signal variables.

It can also be seen that the higher the number of true signal variables in the dataset $p_{\mathcal{P}}$, the more variables entered the final model ($p_{\hat{\mathcal{P}}}$). In this case, the behaviour displayed is in line with what is expected.

Preconditioned models that use NSC in the second phase included many more variables than their PLR counterparts. and only LDA-NSC and LogReg-NSC reduced the number of variables as the signal-to-noise ratio dropped.

The non-preconditioned models had a comparable number of variables to the preconditioned models that use PLR in the second phase.

Overall it seems that NSC-PLR , KNN-PLR, SPC-PLR, PLR and SPC included the smallest number of variables.

In most cases the significant increase in $p_{\hat{\mathcal{P}}}$ when using NSC as the second model, as opposed to PLR, overshadows the slight improvement of error rates. As with the first simulation study, it should be noted that although NSC, LogReg-NSC and LDA-NSC had some success with simultaneous accuracy and dimension reduction in some scenarios, the number of variables included in the model showed very high variance over the 10 iterations. The variances of the number of variables included in the final model are presented and discussed in Appendix A, Figure A.5. Further observations from these results are that as r decreases, the variance of $p_{\hat{\mathcal{P}}}$ for NSC, LogReg-NSC and LDA-NSC first increases before decreasing. For these models, lower values of N generally lead to higher variances. Lastly it can be noted that the variances of the numbers of variables retained in each of the models in the green set seemed relatively low.

Stellenbosch University https://scholar.sun.ac.za

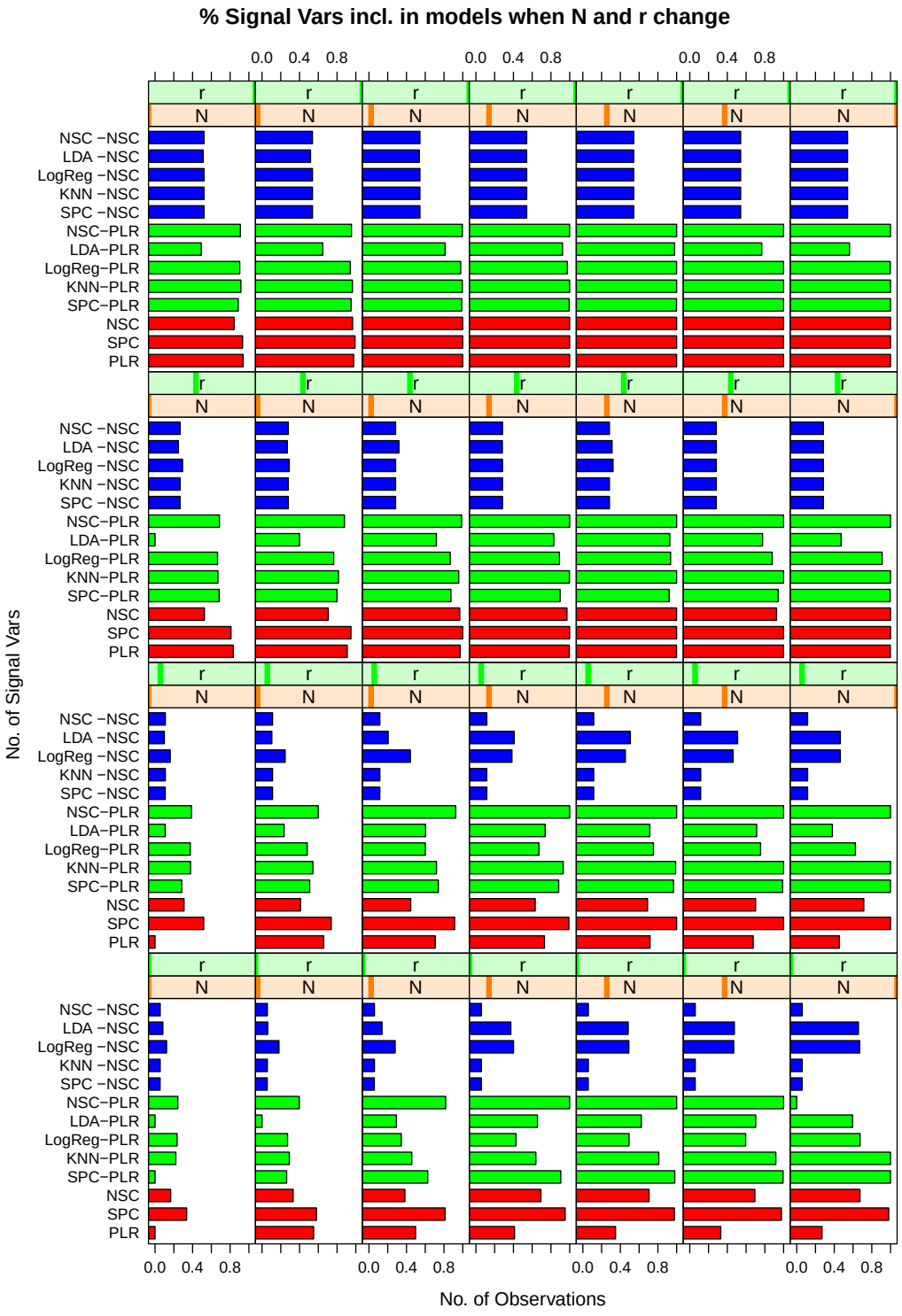

Figure 3.10: The graph above shows the proportion $\mathcal{G}$. The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Average proportion of 'good predictors' ($\mathcal{G}$)
In the first two rows of Figure 3.10 it appears that for a large or moderate r, the size of N does not have much of an impact on the average $\mathcal{G}$. The only models affected by the change in N are LDA-PLR and, to a lesser extent, NSC.

In the bottom two rows, where r is quite low, an increase in N generally leads to an increase in $\mathcal{G}$. However, in the lower right-hand panel, it can be seen that NSC-PLR is an exception to this statement.

The preconditioned models that use PLR in the second phase have a comparable performance, in terms of signal captured, to the non-preconditioned models. As the signal-to-nose ratio decreases, the preconditioned models that use PLR in the second phase gain the advantage.

For the preconditioned models that use NSC in the second phase, the proportion of signal captured starts out fairly decently at the top, but declines as the signal-to-noise ratio decreases. Only LDA-NSC and LogReg-NSC maintain an adequate proportion (as long as N remains fairly high).

The models with the best performance in this regard include: SPC-PLR, KNN-PLR, NSC-PLR and SPC.

The variances of $\mathcal{G}$ are presented and discussed in Appendix A, Figure A.6. From this graph it can be stated that the combination of a smaller r with a larger N resulted in an increase in variance for $\mathcal{G}$. In addition, if the variance for $p_{\hat{\mathcal{P}}}$ was high, then the variance for $\mathcal{G}$ was also high. LDA-PLR and LogReg-PLR seem to produce more variable results even though their corresponding $p_{\hat{\mathcal{P}}}$'s have low variances.

Conclusion
In this section, the main overall results of Simulation Study 2 are summarised.

Firstly, it is interesting to note that in Simulation Study 1, one of the overall best models was NSC-PLR; however, in this simulation study, it becomes clear that it does not always perform admirably. It was the only model with a high error rate when r was small and N was large. The corresponding $\mathcal{G}$ was also low, highlighting the fact that the model was unable to identify the good predictors in this case. It remains a useful model though in cases where N is smaller.

Again, all the models in the blue group had relatively low error rates, although they were much less adept at reducing the dimensionality and are therefore not well suited for the purpose of preconditioning. This is also reflected in their

Stellenbosch University https://scholar.sun.ac.za

relatively low values for $\mathcal{G}$. Although LDA-NSC and LogReg-NSC achieved lower $p_{\hat{\mathcal{P}}}$'s and higher $\mathcal{G}$'s, these were accompanied by large variances, implying that the results are unstable.

Although the green group of models had relatively high error rates when r was large, but N was small, they were able to significantly reduce the dimensionality of the problem and simultaneously achieve a high $\mathcal{G}$. LogReg-PLR, KNN-PLR and SPC-PLR behaved similarly and had fair average error rates, low average $p_{\hat{\mathcal{P}}}$ and high values for $\mathcal{G}$. SPC-PLR performed especially well in this regard for larger values of N.

NSC achieved relatively low average error rates for many of the scenarios and managed to deliver a fairly low $p_{\hat{\mathcal{P}}}$. When both N and r are small, most models fail to identify a large proportion of the signal variables; however, in this case NSC is one of the higher ranked models. The increase in accuracy is paired with a relative increase in $p_{\hat{\mathcal{P}}}$ in many of the scenarios. Even with this increase however, it still produces fairly high values of $\mathcal{G}$. NSC is a fairly robust method over the various scenarios in this simulation study, although its results can have a high variance.

SPC achieved error rates comparable to that of NSC in nearly all the scenarios, and in addition it produced lower $p_{\hat{\mathcal{P}}}$'s than NSC in nearly all the scenarios. This also resulted in a higher $\mathcal{G}$ than that produced by NSC and without the high variance seen in NSC. When only one feature is retained in the predictive model, interpretation of an SPC model is possible.

PLR as a standalone method had average error rates and average $p_{\hat{\mathcal{P}}}$ comparable to those of the green set of models; however, the average value of $\mathcal{G}$ was lower than those obtained by some of the green set of models when r is low. This implies that although this technique is simpler to implement and generally performs as well as preconditioned models that use PLR in the second stage, there are scenarios where it will be beneficial to opt for a preconditioned model instead. SPC-PLR or KNN-PLR can be used instead in cases where r is low.

Finally, taking all three measures into consideration the models providing the best balance include, in no particular order, SPC, PLR, SPC-PLR and KNN-PLR, although NSC is also useful when accuracy is more important than dimension reduction.

Stellenbosch University https://scholar.sun.ac.za

3.4 Final Remarks

The models that were highlighted in the first study were NSC-PLR, SPC, as well as PLR. In the second study, SPC, PLR, SPC-PLR and KNN-PLR were highlighted.

From the above two studies it can be concluded that for binary classification, SPC and PLR have performances comparable to that of a preconditioned PLR model using SPC, NSC or KNN as the preconditioner.

Stellenbosch University https://scholar.sun.ac.za

Chapter 4

Real-World Experiments

In this chapter the models used in the previous chapter are applied to three real-world datasets. The results are depicted and the performance of the models compared to one another as well as to the results obtained by other researchers.

The datasets chosen for this chapter are all gene-expression microarrays with a binary outcome and a fair balance between the classes.

In order to isolate the impact of preconditioning, other preprocessing techniques were limited to only what was necessary, namely excluding variables with zero variance and centring and scaling the data. Other attempts at classifying the data in the literature, first transform the variables to logarithmic scale or exclude variables with a relatively low variance. Although these may lead to improved accuracy or smaller subsets of variables, these approaches were not followed so as to minimise the impact of subjectivity on the analysis.

As before, please note that where reference is made to $p_{\hat{\mathcal{P}}}$ in an SPC context, it refers to the number of variables that survive the screening process, not the number of PCs that were included in the predictive model.

4.1 Breast Cancer

A dataset was obtained from The Bioinformatics Laboratory (Breast Cancer, no date), although the original dataset was collected and analysed by Chang *et al.* (2003). It contains measurements on $p = 12\,625$ genes from $N = 24$ patients who received the same specific type of treatment for breast cancer. Fourteen of the patients were resistant to the treatment, whereas ten of the patients were sensitive towards it. We therefore have $N_1 = 14$ and $N_2 = 10$.

The task is to predict whether a patient with breast cancer will be resis-

Stellenbosch University https://scholar.sun.ac.za

tant or sensitive towards the aforementioned treatment option, based on their genomic profile. Ideally, a small subset of these genes can be identified as the probable cause of the resistance or sensitivity.

Chang *et al.* (2003) used a compound covariate predictor method to obtain a final model containing 92 of the predictors. They performed leave-one-out cross-validation on 18 of the observations and achieved 88% accuracy in the classification problem (based on 6 test cases).

Breast Cancer (no date) obtained 73.33% classification accuracy by projecting the data onto the subspace created by the 8 genes that produced the best average accuracy over 10-fold cross-validation. A technique called VizRank's projection search was used to identify the subset of 8 genes used in their analysis (Demšar *et al.*, 2007). It is unclear which classifier was used to obtain the prediction results, or whether the results refer to the CV-error or the test error.

In this thesis, the dataset was randomly partitioned so that approximately a quarter of the resistant observations and a quarter of the sensitive observations were kept separately as a test set. In total 7 observations were used for the test set: 3 sensitive, 4 resistant. The remaining 17 cases were used as the training set.

Ten of the genes had zero variance and those were removed from the training and test sets. The remaining 12615 variables were centred and scaled to unit variance. No missing values were present in the data.

To obtain a two-dimensional representation of this high-dimensional dataset, supervised principal components was used and the two components most correlated with the response were plotted. The result is displayed in Figure 4.1. The classes seem to be (non-linearly) separable based on this lower dimensional projection.

Stellenbosch University https://scholar.sun.ac.za

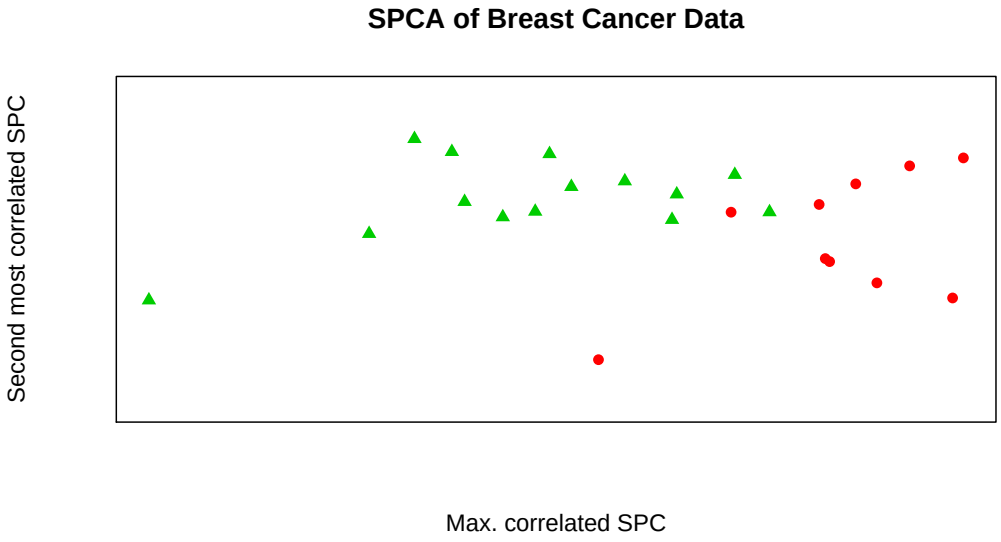

Figure 4.1: A two-dimensional display representation of the breast cancer dataset, obtained by plotting two supervised principal components.

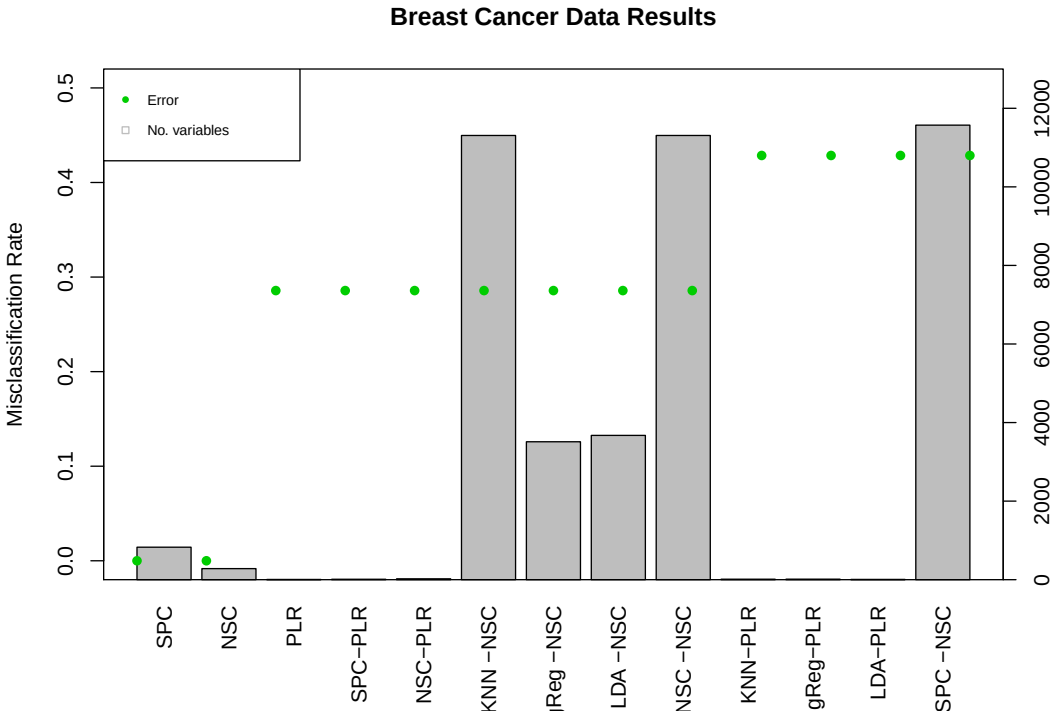

Figure 4.2: In the graph above, various models were applied to the dataset. For each model, the misclassification rates on the test data are represented by the green dots, while the $p_{\hat{\mathcal{P}}}$'s are represented by the grey bars.

Stellenbosch University https://scholar.sun.ac.za

To obtain predictions for this classification problem, 5-fold cross-validation is used to train each of the 13 models introduced in the previous chapter. The results obtained by these models, when evaluated on the test set, are presented in Figure 4.2. The misclassification rates are given by the green dots (left axis) and the number of variables included in the final model are represented by the bars (right axis).

Supervised principal components and nearest shrunken centroids performed well, with 0 misclassifications and producing reduced subsets of 827 and 282 respectively. Although their subset sizes are fairly low compared to the other models, neither of the subsets are small enough to interpret the model with ease.

Although PLR and the preconditioned-PLR models had higher error rates, they managed to select much smaller subsets of predictors. PLR resulted in the lowest subset size of 6 with SPC-PLR and NSC-PLR having subset sizes 11 and 22 respectively. Although KNN-PLR and LogReg-PLR produced small subsets of sizes 12 and 13, their error rates were too large to compete with the other models.

The preconditioned-NSC models mainly resulted in moderate to large subsets of selected predictors and therefore are not well suited to the purpose of this study.

The above findings are in line with the insights gained from the simulation study. In this application, standard models outperform preconditioned models in terms of accuracy, but not in terms of interpretability.

It can also be seen that, in terms of accuracy, the majority of the models presented here were able to achieve greater accuracy than the models of Breast Cancer (no date). These models also retained a smaller subset of predictors than that of Chang *et al.* (2003); however, the subset size used by Breast Cancer (no date) was smaller than those of the preconditioned models in most cases.

Stellenbosch University https://scholar.sun.ac.za

4.2 Prostate Cancer

A second dataset was obtained from The Bioinformatics Laboratory (Prostate, no date), although the original dataset was collected and analysed by Singh *et al.* (2002). It contains measurements on $p = 12\,533$ genes from $N = 102$ samples of prostate tissue. Fifty of the samples are of normal tissue adjacent to a prostate cancer, while 52 samples are prostate tumours. We therefore have $N_1 = 50$ and $N_2 = 52$.

The task is to predict whether a tissue sample is normal or tumorous, based on its genomic profile. Ideally a small subset of these genes can be identified as the probable distinguishing factors.

Singh *et al.* (2002) used two KNN models containing 4 and 16 genes respectively. They performed leave-one-out cross-validation on all the observations of this dataset to determine which genes lead to the lowest cross-validation error. Once the subset was identified, they evaluated the performance on an independent set of 35 observations provided by a different source. These models achieved 77% and 86% accuracy respectively.

Prostate (no date) obtained 85.36% classification accuracy by projecting the data onto the subspace created by the 8 genes that produced the best average accuracy over 10-fold cross-validation. Again, VizRank's projection search was used to identify the subset of 8 genes used in their analysis; however, it is unclear what classifier was used to obtain the results, or whether the results refer to the CV-error or the test error.

In this thesis, the dataset was randomly partitioned so that approximately a quarter of the normal samples and a quarter of the tumorous samples were kept separately as a test set. In total 26 samples were used for the test set: 13 normal samples and 13 tumorous. The remaining 76 samples were used as the training set.

None of the genes had zero variance and there were no missing values present in the data. All columns were however centred and scaled to unit variance.

To obtain a two-dimensional representation of this high-dimensional dataset, supervised principal components was used and the two components most correlated with the response were plotted. The result is displayed in Figure 4.3. Although a perfect separation is not possible, a linear decision boundary could perform fairly well.

Stellenbosch University https://scholar.sun.ac.za

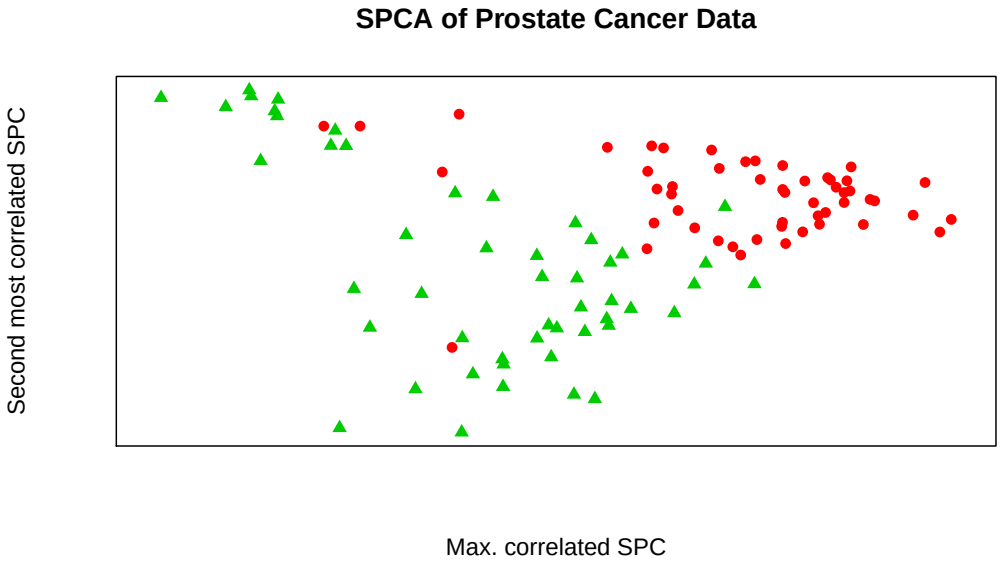

Figure 4.3: A two-dimensional display representation of the prostate cancer dataset, obtained by plotting two supervised principal components.

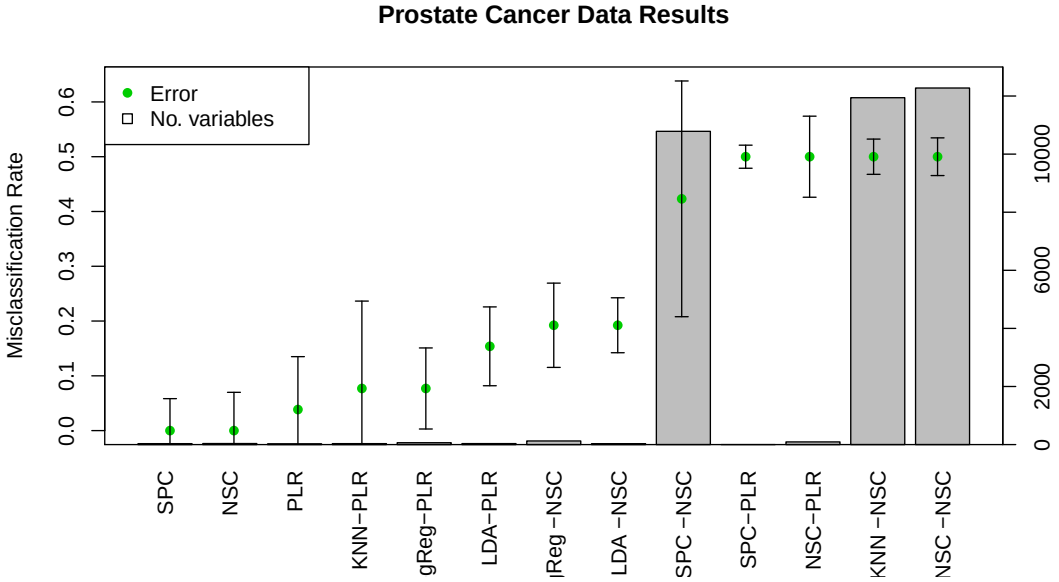

Figure 4.4: In the graph above, various models were applied to the dataset. For each model, the misclassification rates on the test data are represented by the green dots, while the $p_{\hat{P}}$'s are represented by the grey bars.

Stellenbosch University https://scholar.sun.ac.za

The same preprocessing and models described for the breast cancer dataset were used in order to obtain predictions of the prostate sample types. The results obtained by these models, when evaluated on the test set, are presented in Figure 4.4. The misclassification rates are given by the green dots (left axis) and the number of variables included in the final model are represented by the grey bars (right axis).

In addition, 5 different random splits between the training and test set were considered so as to obtain estimates of the standard deviations of the results. The one-standard deviation error bars for the error rate are depicted by the black lines on the graph. Although not depicted, here, the standard deviation of the number of variables included in the final model was also estimated. These were relatively low for all models, except for SPC-NSC. This might also explain the large variability of its error rate. Since the size of the test set is larger than for the breast cancer application, the misclassification rates seem more like a continuous function than a step function over the different models.

Again, supervised principal components and nearest shrunken centroids were able to classify the test cases perfectly. In this application they were also able to identify a relatively smaller subset of genes. The sizes of the subsets identified by the two techniques are 32 and 36 respectively.

Three of the preconditioned NSC models, namely SPC-NSC, KNN-NSC and NSC-NSC, were not only unable to reduce the dimensionality of the problem noticeably, but they also performed poorly on the test set. The models SPC-PLR and NSC-PLR reduced the dimensionality of the problem (with SPC-PLR only retaining an intercept term); however, their performances on the test set were also close to that of random guessing.

Of the preconditioned models, KNN-PLR and LogReg-PLR had the lowest misclassification rates. In addition, these had $p_{\hat{\mathcal{P}}} = 33$ and $p_{\hat{\mathcal{P}}} = 64$ respectively. The standard deviation for KNN-PLR was relatively high, however, and therefore LogReg-PLR might be considered the better of the two.

In this case, standard models seem to provide the best balance of both criteria. In addition, they all displayed a low standard deviation of error rates.

In terms of accuracy, some of the models presented here were able to achieve greater accuracy than the models of Singh *et al.* (2002) and Prostate (no date). These models also retained a smaller subset of predictors than those of Singh *et al.* (2002), although the subset size used by Prostate (no date) is smaller in all cases.

Stellenbosch University https://scholar.sun.ac.za

4.3 Medulloblastoma

A third dataset was obtained from The Bioinformatics Laboratory (Medulloblastoma, no date), although the original dataset was collected and analysed by MacDonald *et al.* (2001). It contains measurements on $p = 1\,465$ genes from $N = 23$ samples of prostate tissue. Ten of the samples are of metastatic medulloblastomas, while 13 samples are of non-metastatic medulloblastomas. We therefore have $N_1 = 10$ and $N_2 = 13$. Medulloblastoma is the most common malignant brain tumour of childhood. A metastatic tumour is one which has spread from the primary site of origin into a different part of the body. This particular disease tends to spread throughout the central nervous system early in the course of the illness, which makes treatment complicated (MacDonald *et al.*, 2001).

The task here is to predict whether a medulloblastoma is metastatic or non-metastatic, based on its genomic profile. Ideally, a small subset of these genes can be identified as the probable distinguishing factors.

MacDonald *et al.* (2001) performed leave-one-out cross-validation to arrive at a predictive model containing 85 predictors and achieving a 72% CV-accuracy. They tested their model on 4 independent observations, which were all correctly classified.

Medulloblastoma (no date) obtained 51.67% classification accuracy by projecting the data onto the subspace created by the 8 genes that produced the best average accuracy over 10-fold cross-validation. Again, VizRank's projection search was used to identify the subset of 8 genes used in their analysis; however, it is unclear what classifier was used to obtain the results or whether the results refer to CV-error or test error.

In this thesis, the dataset was randomly partitioned so that approximately a quarter of the metastatic cases and a quarter of the non-metastatic cases were kept separately as a test set. In total 7 cases were used for the test set: 3 metastatic cases and 4 non-metastatic. The remaining 16 cases were used as the training set.

None of the genes had zero variance and there were no missing values present in the data. All columns were however centred and scaled to unit variance.

To obtain a two-dimensional representation of this high-dimensional dataset, supervised principal components was used and the two components most correlated with the response were plotted. The result is displayed in Figure 4.5. Some overlap is present in the data, but on the whole, the groups are fairly distinct from one another.

Stellenbosch University https://scholar.sun.ac.za

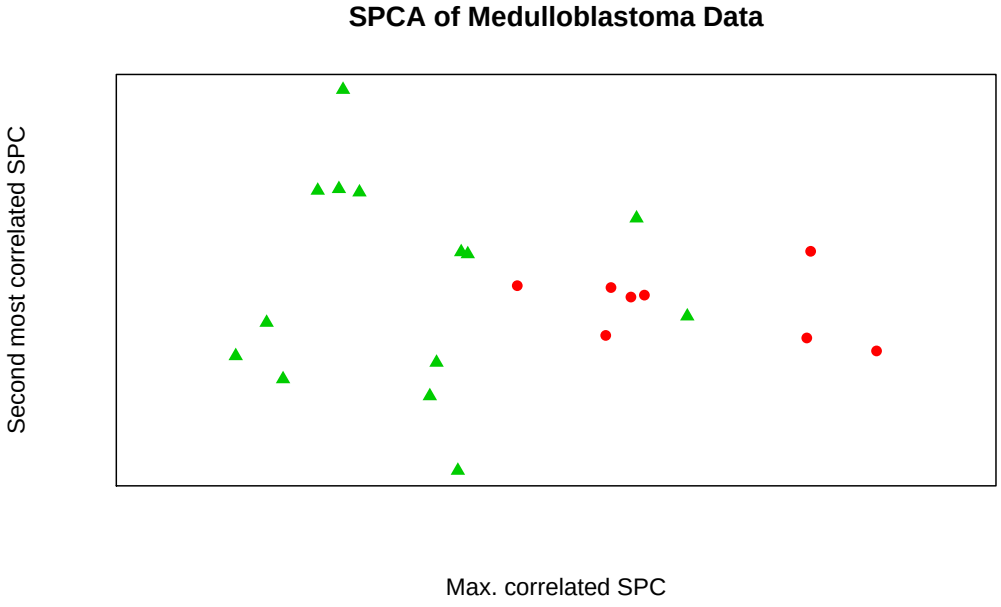

Figure 4.5: A two-dimensional display representation of the medulloblastomas dataset, obtained by plotting two supervised principal components.

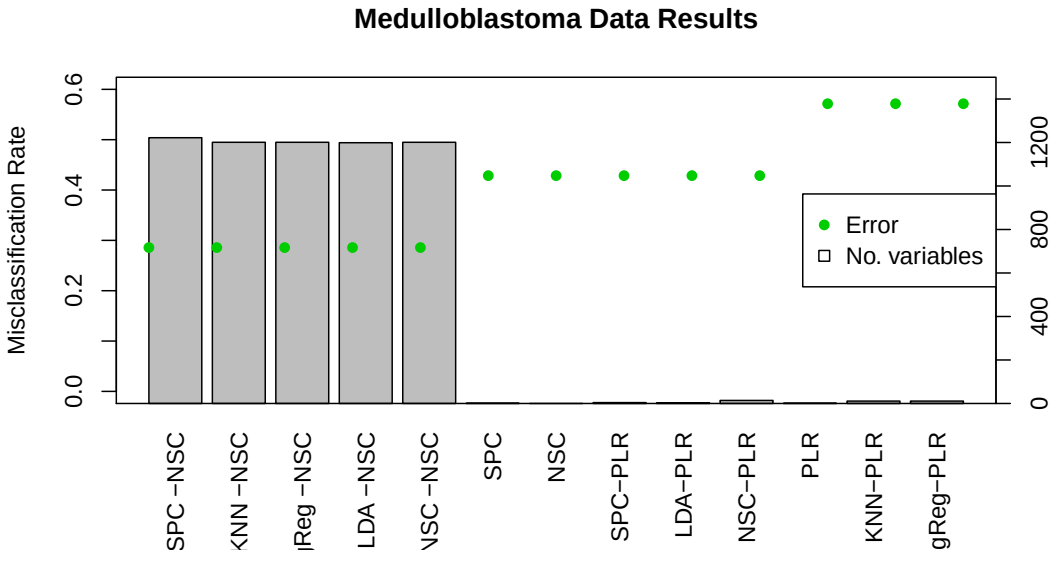

Figure 4.6: In the graph above, various models were applied to the dataset. For each model, the misclassification rates on the test data are represented by the green dots, while the $p_{\hat{\mathcal{P}}}$'s are represented by the grey bars.

Stellenbosch University https://scholar.sun.ac.za

The same preprocessing and modelling techniques described for the previous 2 datasets were used in order to obtain predictions of the medulloblastomas. The results obtained by these models, when evaluated on the test set, are presented in Figure 4.6. The misclassification rates are given by the green dots (left axis) and the number of variables included in the final model are represented by the bars (right axis). Due to the small sample size and the resulting small test set, the error rates follow a step function pattern.

Contrary to the previous results, the best error rate performance came from the preconditioned-NSC models. These models however, used noticeably more variables than the other models.

Of the preconditioned-PLR models, SPC-PLR, LDA-PLR and NSC-PLR were on par with SPC and NSC in terms of misclassification rates. SPC selected only 1 predictor, NSC only an intercept term and the models SPC-PLR, LDA-PLR and NSC-PLR selected 4, 3 and 14 predictors respectively.

The remainder of the models performed worse than random guessing, with misclassification rates on the test set around 57%. This inaccuracy may be due to the small sample size.

Another noteworthy aspect is that in this instance, SPC-PLR, LDA-PLR and NSC-PLR outperformed KNN-PLR and LogReg-PLR, whereas the reverse was true for the previous two examples.

In terms of accuracy, the majority of the models presented here were able to achieve greater accuracy than the model of Medulloblastoma (no date) and in these cases the subset of predictors retained was also mainly smaller than 8. MacDonald *et al.* (2001) achieved the greatest accuracy however. Although the subset of predictors they retained is larger than that of the standard models and the preconditioned-PLR models in this thesis, 85 is still much lower than the subset sizes produced by the preconditioned-NSC models.

4.4 Concluding Remarks

The three real world applications revealed that there is no universal strategy that is best in all cases. The standard algorithms performed well in the first 2 scenarios, having both low error rates and a sparse final model. Of the preconditioned algorithms, preconditioned-PLR models also offered a good balance between accuracy and sparsity. Members of this group of models could obtain comparable or better error rates than those of the preconditioned-NSC models, while only retaining a fraction of the variables. In these two applications, the

Stellenbosch University https://scholar.sun.ac.za

models presented here either outperformed, or were comparable to the results found on these datasets in the literature.

In the final example, however, preconditioned-NSC models resulted in a much lower error rate, but included many more variables into the model, whereas the sparser models yielded higher error rates. The standard models therefore did not outperform preconditioned-NSC in terms of accuracy in this case, as it did in the first two cases. It did however lead to more interpretable solutions and therefore might still be considered to provide the best balance of accuracy and interpretability. The relatively low accuracy rate of the model used by Medulloblastoma (no date) indicates that this case represents a more challenging classification problem than those considered in the first two examples.

On balance, it seems that SPC and NSC consistently performed at least reasonably well. It therefore appears that these models are more stable, or robust, over different scenarios than preconditioned models.

Stellenbosch University https://scholar.sun.ac.za

Chapter 5

Concluding Remarks and Further Research

In this chapter the main concept from each chapter is summarised, as well as the main findings from the study. A few possibilities for further research are also indicated.

5.1 Concluding Remarks

In Chapter 1 key concepts and terminology surrounding supervised learning and unsupervised learning were discussed. In particular, this chapter included topics such as loss functions, prediction errors and error rates, overfitting, the bias-variance trade-off, as well as the curse of dimensionality and the difficulty of overcoming it.

In Chapter 2 certain techniques and models within the fields of variable selection, shrinkage methods and feature selection were discussed. Preconditioning was presented as a technique that can result in dimension reduction, as well as accurate and interpretable solutions when applied to high-dimensional data.

In Chapter 3 a simulation study was designed and carried out, testing the behaviour of preconditioned models under specific circumstances. The study also compares the performances of these models relative to the performance of other standard (non-preconditioned) techniques and relative to variants of preconditioning itself. Performance was measured in terms of three criteria, namely the misclassification rate on the test set, the number of variables included in the final model and what proportion of these variables are signal bearing (as defined in the setup of the simulated datasets).

In Chapter 4 the models were applied to three actual datasets in the field of genomics and their performances were compared.

Stellenbosch University https://scholar.sun.ac.za

Analysis is problematic when high-dimensional data are involved. Regularisation and dimension reduction are two tools that can be used to alleviate the associated problems. Certain techniques, such as SIS, focus on reducing the number of predictors as a preprocessing step, before a predictive model is applied. Others focus on reducing dimensionality whilst training the predictive model. These only consider predictors that have a significant impact on the accuracy of the model, or alternatively penalise coefficients while training the model. These methods perform poorly in very high-dimensional cases however. Preconditioning is a two-stage technique that aims to reduce the noise inherent in the training data before making final predictions, so as to improve the performance of dimension reduction techniques. This is done by a preprocessing step whereby predicted response values are produced. These are then paired with the original input variables to use as new training data-pairs to fit a final prediction model. The literature of this technique focuses on the regression case, but in this thesis, the technique is applied to a classification setting.

In the simulation studies performed, preconditioned models (using different combinations of predictive models in the first stage and in the second stage) were compared amongst themselves, as well as to standard, non-preconditioned models. The results were compared over various scenarios involving the structure of the simulated data, based on three performance measures.

In general it was found that preconditioned-NSC models do not perform well in terms of interpretability measures, but may produce accurate predictions. Preconditioned-PLR regression models on the other hand have lower accuracy in general compared to the preconditioned-NSC models, but their significant improvement in interpretability over the preconditioned-NSC models outweighs the reduction in accuracy in many of the scenarios.

The standard models, namely NSC, SPC and PLR, had relatively low misclassification rates. Often NSC was the most accurate model, although its drawback is that it does not reduce the dimensionality of the problem sufficiently. SPC and PLR provide good accuracies and reduces the dimensionality of the problem sufficiently, therefore providing interpretable results. Although the standard techniques generally perform well (with less computation), there are scenarios under which preconditioned-PLR classification models can achieve comparable accuracy to that of standard classification techniques, while remaining both interpretable and sparse.

Application of the models to the real-world datasets confirmed that there is no universally dominant technique, although SPC and NSC consistently performed at least reasonably well, whereas the same conclusion cannot be made

Stellenbosch University https://scholar.sun.ac.za

about the other models.

Therefore, based on the results presented in this document, it appears that, at their best, preconditioned classifiers can only reach a performance that is on par with standard classifiers. Preconditioned classifiers can however, also perform much worse.

5.2 Suggestions for Future Research

Supervised principal components is a fairly simple technique, which performed well in many of the scenarios, in terms of accuracy and interpretability. As mentioned in Section 2.4.5, this technique falls into a broader framework of features formed by PCs that result from a PCA that is performed on a weighted form of the input matrix. Y-aware PCA also falls within this framework, but the framework leaves room for more possibilities of weight matrices that could be applied. Two suggestions for such a matrix were made in Section 2.4.5, but no theoretical or empirical work was done to determine the viability of these suggestions. The results arising from such an analysis could be compared against SPC in terms of lower-dimensional visualisations of the data, as well as the use of the PCs arising from these methods, as features in a predictive model.

Related to preconditioned models, an improvement which could be considered when tuning the parameters for preconditioned algorithms, is to incorporate a metric related to the number of variables in the final model. Due to the fact that preconditioning has a dual purpose to fulfil (generalises well and sparsity), the measure by which parameters are optimised in cross-validation needs to be a composite of not only some measure of error, but also of the number of variables included in the model. Using only the error rate can lead to sparsity being downplayed in favour of a slight improvement in accuracy, as can be seen in the case of the preconditioned models that use NSC in the second phase.

Stellenbosch University https://scholar.sun.ac.za

Appendices

Stellenbosch University https://scholar.sun.ac.za

Appendix A

Further Results

A.1 Study 1

The following graph shows the variances of the misclassification error rates on the test sets over the 10 iterations of Simulation Study 1. The variances here should be viewed alongside Figure 3.5.

The variances of all techniques are the lowest in the lower right-hand corner of the graph, where the groups are well separated in terms of means and variances.

In the first column of the graph, it can be seen that SPC-PLR had the lowest variance in error rates, while SPC had among the highest. The green group of models seems to have lower variances on average than models in the other two groups. In this case, since the average error rates were around 0.5 for all the models, a higher variance could lead to improved results, at least for some of the time.

In column 2 of the plot, this is slightly more evident. From top to bottom, the average test error was high in the first 3 rows, but decreased steadily in the bottom two panels. In the plot of the variances however, the pattern is reversed. It was stated in Chapter 1 that the bias-variance trade-off does not extend to the classification case, since when predictions are bad, an increase in variance could lead to improved results. This phenomenon is what is driving the results in this column of the plot. Of course, when predictions are accurate, a low variance is preferable, as an increase in variances will impact negatively on accuracy. This statement is supported by the results of the final column of the plot. Those models with very low misclassification rates also had low variances, whereas LDA-PLR had a relatively higher error rate, as well as variance.

Stellenbosch University https://scholar.sun.ac.za

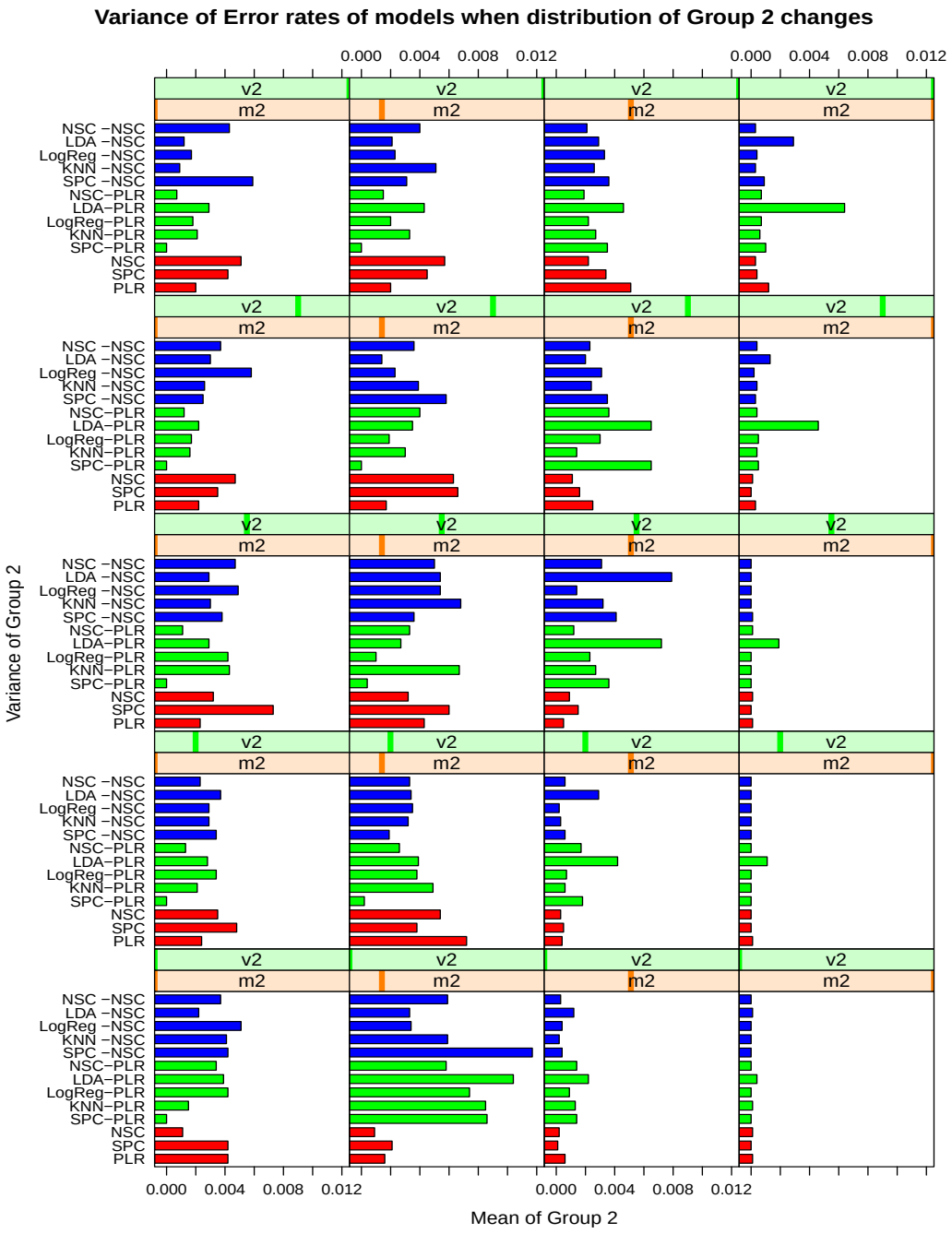

Figure A.1: The graph above shows the variances of the misclassification rates on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 increases from left to right, while the variance decreases from top to bottom.

It is in column three of the plot where the relative variances of the models become clearer. NSC consistently has a relatively low variance in the error rates.

Stellenbosch University https://scholar.sun.ac.za

As v2 decreases, the the variance of the error rates produced by LDA-NSC increases relative to the variances of the other models' performances, whereas the variance of PLR decreases relative to that of other models. In general, the patterns of bars produced in column three and four of this chart look strikingly similar to the pattern produced by the average error rate bars in Figure 3.5.

The following graph shows the variances of the number of variables included in the final model presented in Simulation Study 1. The variances here should be viewed alongside Figure 3.6.

From this figure it should be noted that the variances of LDA-NSC, LogReg-NSC and NSC were exorbitant, as can be seen by the scale of the variance in the axis. In comparison, the variances for all the other models seem to be near zero in all the scenarios. This suggests that LDA-NSC, LogReg-NSC and NSC use a more data-driven approach to dimension reduction than the other models, since a change in the sample used as a training dataset affects these models' decisions on how many variables to retain greatly.

Of the models with high variance, it can be seen that poor separability in the classes (upper left-hand corner) lead to higher variances, while well separated classes leads to lower variances. LDA-NSC maintained a comparatively high variance throughout all scenarios. Of the blue group of models, the models exhibiting the largest variances here, were also the models to achieve greater dimension reduction (Figure 3.6). None of the models in the green group had massive variances, suggesting that the model used in the second step of preconditioning may have more of an influence on the number of variables included in the final model than the model used in the first step. Of the red group of models, NSC had a large variance, suggesting that the variance of the models in the blue group may be connected to the NSC technique.

Stellenbosch University https://scholar.sun.ac.za

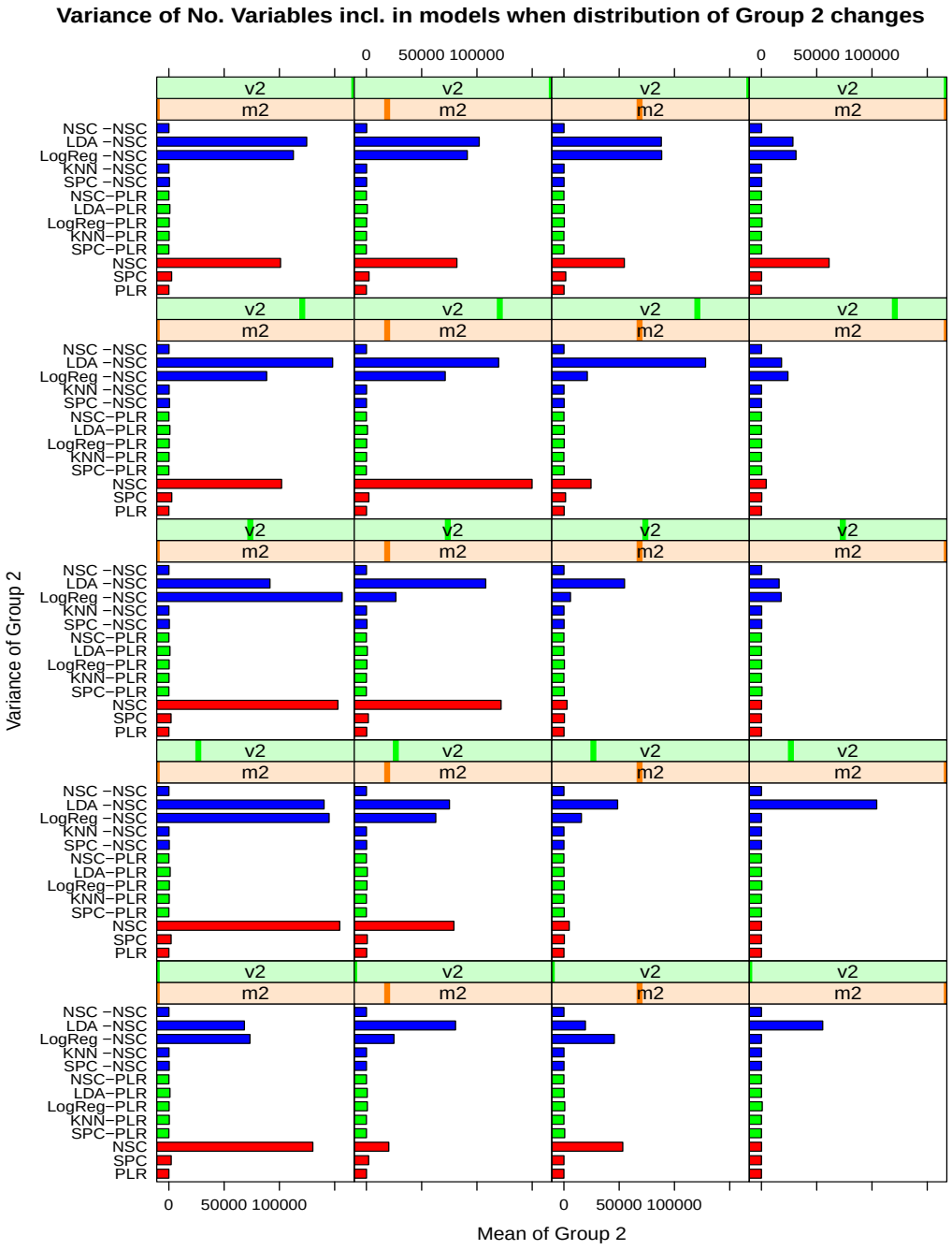

Figure A.2: The graph above shows the variances of the number of variables included in the final model (excluding the intercept term). The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 increases from left to right, while the variance decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

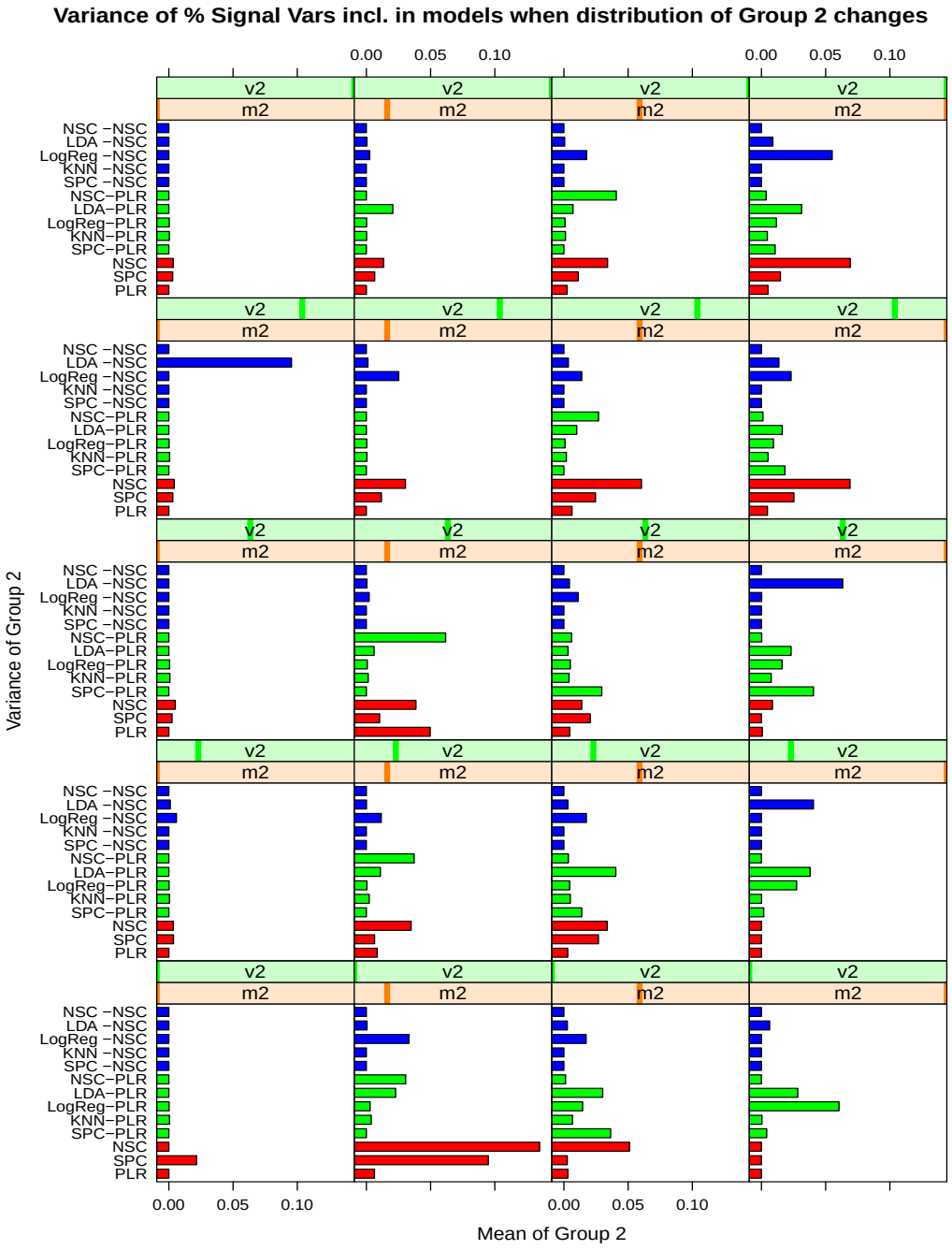

Figure A.3: The graph above shows the variances of the proportion $\mathcal{G}$. The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.3. The mean of Group 2 increases from left to right, while the variance decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

The next graph shows the variances of the proportion of signal variables included in the final model presented in Simulation Study 1. The variances here should be viewed alongside Figure 3.7.

In general, the variances of $\mathcal{G}$ are quite low to the left-hand side of the plot, but they increase as the difference in the means of Group 1 and Group 2 increases (from left to right). Since $\mathcal{G}$ is a function of the number of variables included in the model, a low variance in the number of variables selected will drive a lower variance in $\mathcal{G}$. This can be seen when $p_{\hat{\mathcal{P}}}$ is large, such as in the first column of this plot. In this case the average $\mathcal{G}$ will be low, and so only a very drastic increase in the number of "good predictors" selected will be able to sway $\mathcal{G}$ and therefore increase the variance. It is interesting to note that the variance of LDA-NSC is relatively high, even though the average $\mathcal{G}$ is low. One possibility of why this might be the case is because of the high variance in $p_{\hat{\mathcal{P}}}$, but there must be some other contributing factor, as the other models that exhibited a high variance in $p_{\hat{\mathcal{P}}}$ did not experience the same spike in $var(\mathcal{G})$. The other contributing factor then might be that the model is sometimes good at estimating $\mathcal{P}$, but cannot do so consistently.

Moving to the second and third columns, it seems that the non-preconditioned models have a higher variance, although this is to be expected, since the average $\mathcal{G}$ is also larger. It is however interesting to note that when comparing NSC-PLR with NSC and SPC in the bottom panel of this graph, and considering this same panel in Figure 3.7, it can be seen that although the averages of $\mathcal{G}$ are comparable, $var(\mathcal{G})$ obtained by NSC and SPC is much higher. It seems that the green preconditioned models result in more stable outcomes, for a given average $\mathcal{G}$ than non-preconditioned models.

In the final column, it seems that the variances of the non-preconditioned models, and NSC in particular, decrease along with v2. This is the case for all models where the average $\mathcal{G}$ approaches 1 and implies that the model is fairly certain of its estimate of $\mathcal{P}$. Where the proportion $\mathcal{G}$ was high, but not quite close to 1 yet, and $p_{\hat{\mathcal{P}}}$ was relatively low, the variance of the models increase, since including or excluding a "good predictor" has a higher influence on $\mathcal{G}$.

Stellenbosch University https://scholar.sun.ac.za

A.2 Study 2

The following graph shows the variances of the error rates presented in Simulation Study 2. The variances here should be viewed alongside Figure 3.8.

The variance is lowest where N is large, but moving from left to right over the plot, the variances quickly drop to nearly zero. In the case where r is low, the variances start off higher and tend to zero more slowly.

The green set of models have higher variance compared to the other two sets when r is large, but the variances of these models fall back in line with the rest as r decreases. Some models exhibit a higher variance in misclassification error rates than others, but no model seems to have the highest variance consistently.

When viewed alongside the average error rates, it can be seen that the variances seem to be higher when the average error rates are also high. This is not always the case, as can be seen by the relatively high average error rate of LDA-PLR in the first column and third row of results and the relatively low variance of error rate in the same panel.

Stellenbosch University https://scholar.sun.ac.za

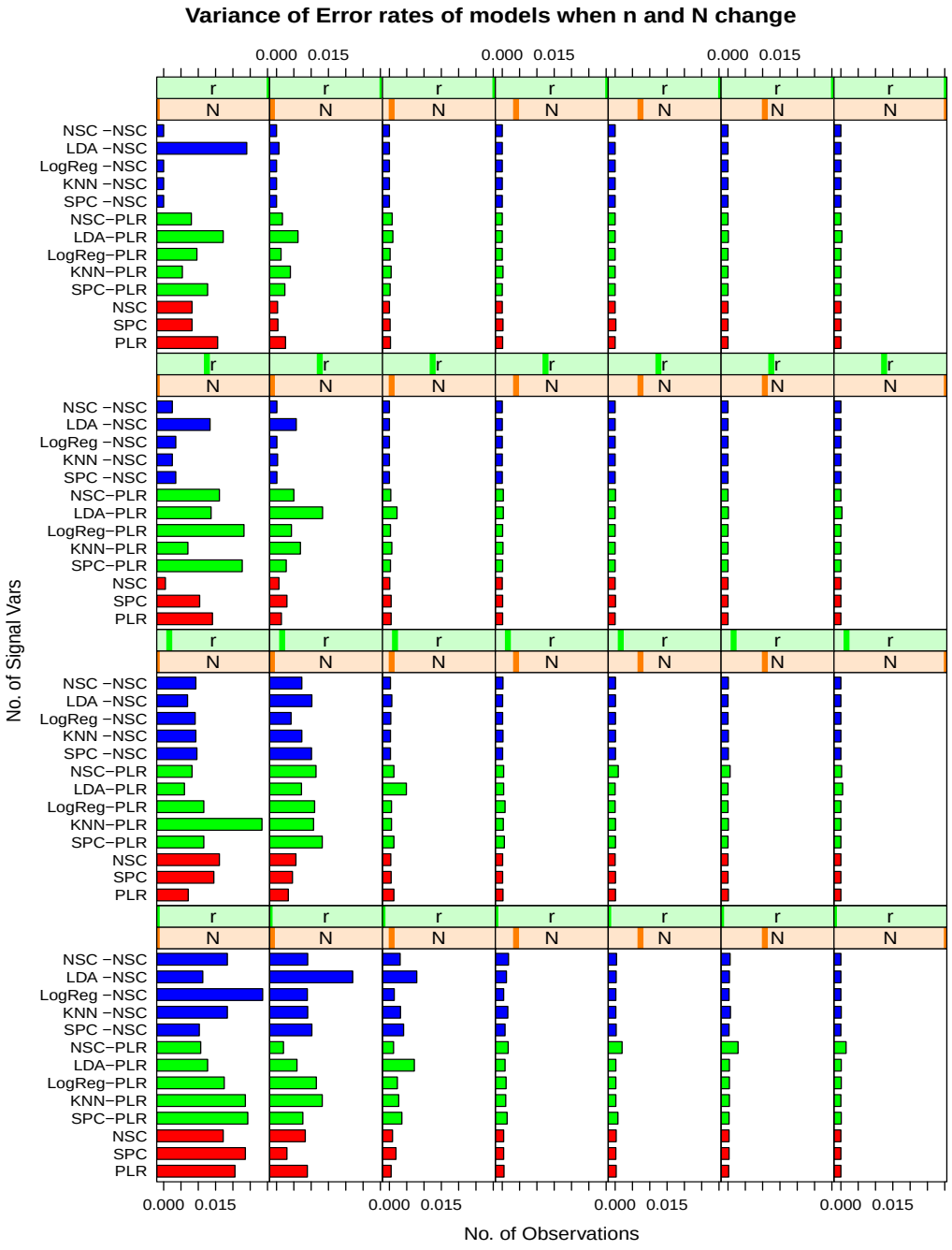

Figure A.4: The graph above shows the variances of the misclassification rates on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Figure A.5 shows the variances of the number of variables included in the final model, $p_{\hat{\mathcal{P}}}$, presented in Simulation Study 2. The variances here should be viewed alongside Figure 3.9.

This graph bears resemblance to Figure A.2 in that there are outlier models with extremely high variances. Again, LDA-NSC, LogReg-NSC and NSC were the models involved in these high variances.

In general one would expect the variance to decrease as N increases, to produce consistent estimators. This is however not the case with LDA-NSC, LogReg-NSC and NSC in the second and third row of panels. The variances of these models decrease up to a point, and then increases again. One would also expect the variance of the results to increase as r decreases. This seems to be the case when observing the outlier models in the first column of results. It is however difficult to draw conclusions surrounding the stability of the results of the other models, since the scale of the graph obscures their performance.

As before, none of the models in the green set produced any outlier variances.

Stellenbosch University https://scholar.sun.ac.za

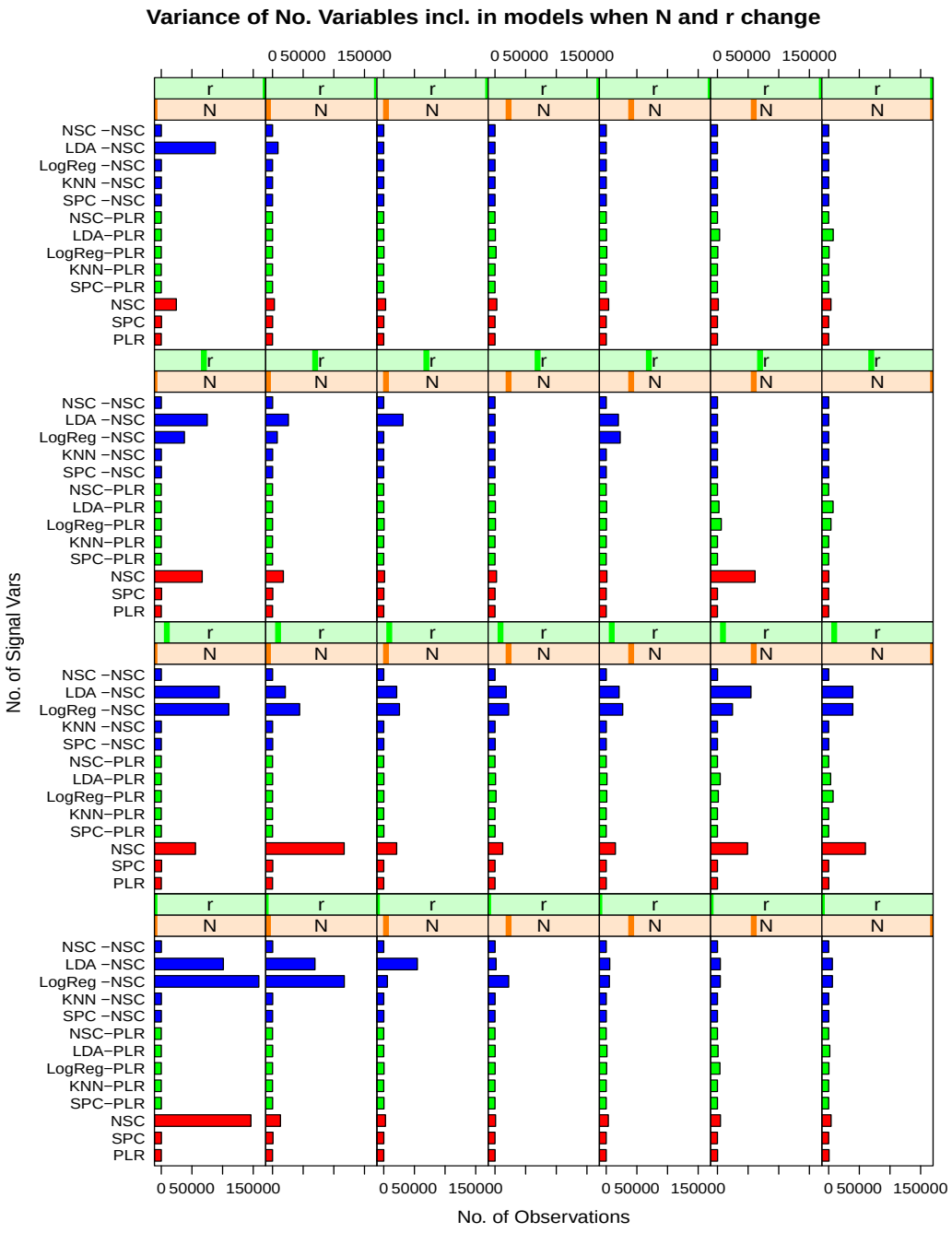

Figure A.5: The graph above shows the variances of the number of variables included in the final model $p_{\hat{\mathcal{P}}}$ (excluding the intercept term). The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

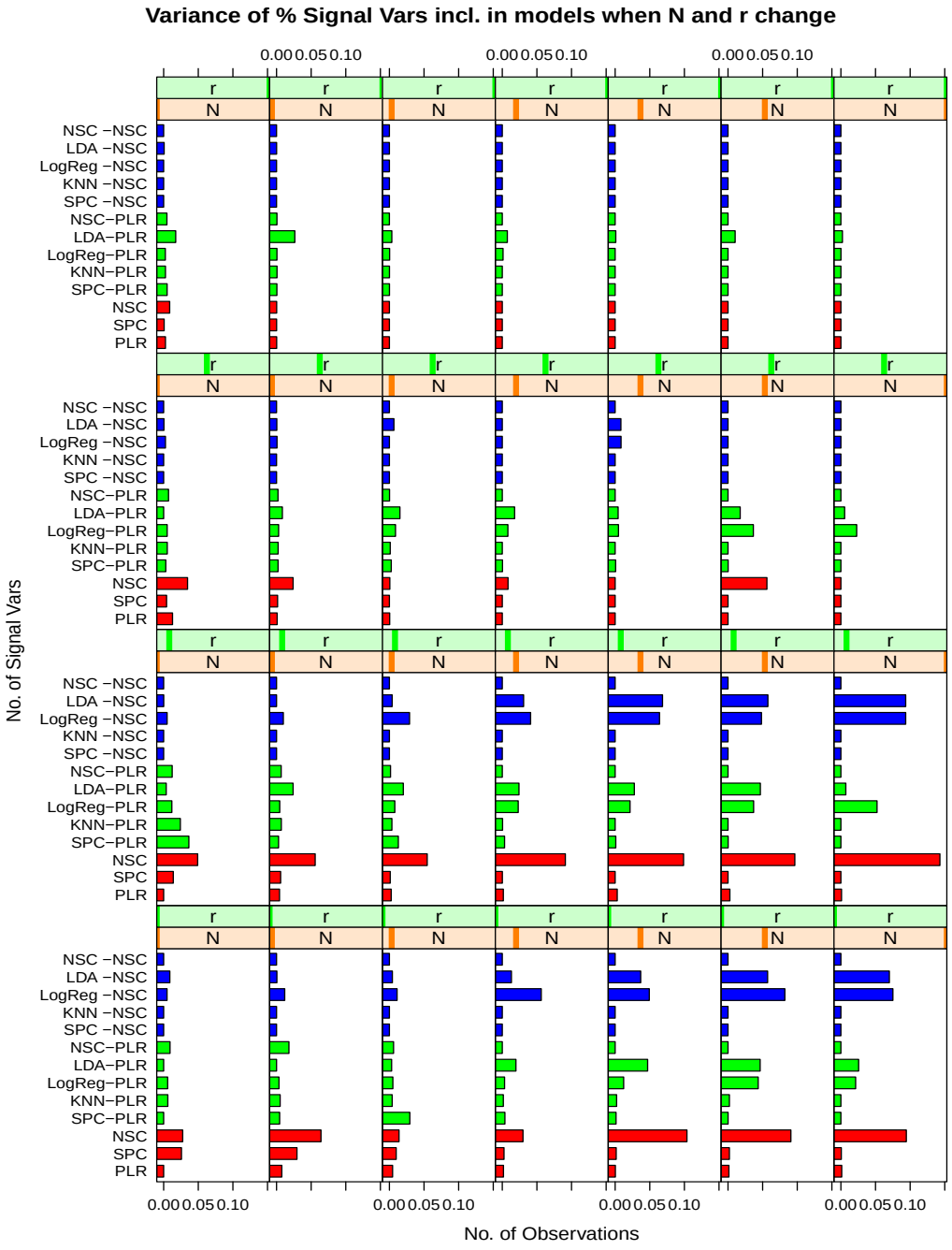

Figure A.6: The graph above shows the variances of the proportion $\mathcal{G}$. The results are based on 5-fold cross-validated models over 10 iterations of data simulated according to the structure of Figure 3.4. The number of observations increases from left to right, while the signal-to-noise ratio decreases from top to bottom.

Stellenbosch University https://scholar.sun.ac.za

Figure A.6 shows the variances of the proportion of signal variables included in the final model presented in Simulation Study 2. The variances here should be viewed alongside Figure 3.10.

In general the variances are low when r is large. As r decreases, some of the variances increase fairly significantly, while others are only marginally affected. The models that are more affected are those with a higher variance of $p_{\hat{\mathcal{P}}}$, namely LDA-NSC, LogReg-NSC and NSC. Again, this is because $\mathcal{G}$ is a function of $p_{\hat{\mathcal{P}}}$.

In addition, the models NSC-PLR and LDA-PLR also produced a relatively higher variance of $\mathcal{G}$. The variance in this case does not stem from the number of variables, but rather from the ability of the model to identify the 'good predictors'.

Stellenbosch University https://scholar.sun.ac.za

Appendix B

Source Code

B.1 Chapter 3 Code: Simulation Study

R Code B.1: Source Code: Generating Data

```
Gen.data<-function (n,p,r,m1=0,m2,v1=1,v2) {
    N1<-floor(n/2)
    N2<-n-N1
    xmat <- matrix(rnorm(p*n,m1,v1), ncol=p, nrow=n)
    xmat[(N1+1):n, 1:r]<-matrix(rnorm(r*N2,m2,v2), ncol=r, nrow=N2)
    yvec = c(rep(0,N1), rep(1,N2))
    data.xy = list(X.mat=xmat, y.vec=yvec)
    return(data.xy)
}
```

R Code B.2: Source Code: Partition Data

```
PartData<-function(X.mat, y.vec){
    N<-nrow(as.matrix(X.mat))
    ind0<-which(y.vec==0)
    set0<-X.mat[ind0,]
    n0<-nrow(set0)
    n0.tr<-floor(0.75*n0)
    set0.tr<-sample((1:N)[ind0], n0.tr, replace=F)

    ind1<-which(y.vec==1)
    set1<-X.mat[ind1,]
    n1<-nrow(set1)
    n1.tr<-floor(0.75*n1)
    set1.tr<-sample((1:N)[ind1], n1.tr, replace=F)
    trainIndex<-sort(c(set0.tr, set1.tr))
    return(trainIndex)
}
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.3: Source Code: Preprocess Data

```
# Preprocess and partition data
prepro<-function(X.mat, y.vec){
    vars<-apply(X.mat,2,var)
    if(min(vars)<0.00001){X.mat<- X.mat[, -which(vars<0.00001)]}
    X.mat<-scale(X.mat)

    if(is.null(colnames(X.mat))){colnames(X.mat)<-paste("x",1:ncol(X.mat))}
    # Partition into train and test
    trainIndex<-PartData(X.mat, y.vec)
    X.train <- X.mat[trainIndex,]
    X.validation <- X.mat[-trainIndex,]
    y.train <- y.vec[trainIndex]
    y.validation <- y.vec[-trainIndex]
    train.set<-cbind(X.train, y.train)

    return(list(X.train=X.train, X.validation=X.validation, y.train=y.train, y
        .validation=y.validation))
}
# library(spls)
# data(prostate)
# prepro(prostate$x, prostate$y)
```

R Code B.4: Source Code: Principal Component Analysis

```
PCA<-function(X.mat, scale=T){
    if(scale==T){X.mat<-scale(X.mat)}
    SVD<-svd(X.mat)
    return(list(V.rotation=SVD$v,UD.princomp=X.mat%*%SVD$v, D.var=SVD$d ))
}
# PCA(wine[,-1])
#
# plot(PCA(prostate$x)$UD.princomp[,1:2])
```

R Code B.5: Source Code: Supervised Principal Components

```
SPCA<-function(X.mat=X.train, y.vec=y.train, gamma=0.1, scale=T){
    score.func<-function(x.vec, y.vec2){
        s<-cor(x.vec,y.vec2)
        return(s)
    }

    if(scale==T){X.mat<-scale(X.mat)}
    scores<-apply(X.mat, 2, function(x) score.func(x, y.vec))

    retain<-abs(scores)>gamma
    retain[which(is.na(retain))]<-F
    if(sum(retain)==0){return(list("Gamma too large"))}
    X.red<-X.mat[,retain]
    PCA.out<-PCA(X.red)
    PCA.out[['retained']]<-retain
    PCA.out[['scores']]<-sort(abs(scores))
    return(PCA.out)
}

# SPCA(wine[,-1], wine[,1], gamma=0.7)
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.6: Source Code: Cross-Validate SPCA

```
cv.SPCA<-function(X.mat, y.vec, k, scale=T){

    options(warn=-1)
    folds<-matrix(rep(1:k), nc=1, nr=nrow(X.mat))
    options(warn=0)

    c.num<-min(ncol(X.mat), max(floor(ncol(X.mat)/10),100))
    Cor<-matrix(0, nc=k, nr=c.num)
    mis<-matrix(0, nc=k, nr=c.num)
    for (j in 1:k){
        # Split non-test part into training and validation
        X.train<-X.mat[folds!=j,]
        X.validation<-X.mat[folds==j,]
        y.train<-y.vec[folds!=j]
        y.validation<-y.vec[folds==j]

        zero.var<-which(round(apply(X.train,2,var), 3)==0)
        if(length(zero.var)!=0){
            X.train<-X.train[,-zero.var]
        }
        Cor[,j]<-sort(abs(apply(X.train, 2, function(x) cor(x, y.train)))
            -0.0001, decreasing = T)[1:c.num]

        for (i in 1:nrow(Cor)){
            result<-SPCA(X.train, y.train, Cor[i,j], scale = scale)
            SPC<-result$UD.princomp
            ret<-result$retained
            spc.cor<-abs(apply(SPC,2, function(pc)cor(pc, y.train)))
            rot<-result$V.rotation
            use.SPC<-which(spc.cor==max(spc.cor))
            LR<-suppressWarnings(glm(y~x, data=data.frame(x=SPC[, use.SPC], y=
                y.train), family = "binomial"))
            X.validation.red<-X.validation[,ret]
            if(all(dim(rot)==c(1,1))){X.valid.rot<- X.validation.red}else{X.
                valid.rot<-X.validation.red%*%rot}
            if(is.matrix(X.valid.rot)){X.new<- X.valid.rot[,use.SPC]}else{X.
                new<- X.valid.rot}
            g<-ifelse(predict(LR, newdata=data.frame(x=X.new), type="response"
                )<0.5, 0,1)
            mis[i,j]<-sum(g!=y.validation)/length(y.validation)
        }
    }
    ave.Cor<-apply(Cor, 1, mean)
    ave.mis<-apply(mis, 1, mean)
    gamma<-ave.Cor[which(ave.mis==min(ave.mis))][1]
    SPCA.final<-SPCA(X.mat, y.vec, gamma, scale = scale)
    while (SPCA.final[[1]][1]=="Gamma too large"){
        gamma<-gamma-0.01
        SPCA.final<-SPCA(X.mat, y.vec, gamma, scale = scale)
    }
    SPCA.final[['Gamma']]<-gamma
    return(SPCA.final)

}
# cv.SPCA(X.mat=X.train_val[,1:100], y.vec=y.train_val, k=5, scale =T)
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.7: Source Code: Variables included by Nearest Shrunken Centroids

```
NSC.Incl.Vars<-function(fit, x, threshold){
    aa <- pamr.predict(fit, x, threshold = threshold, type = "nonzero")
    cen <- pamr.predict(fit, x, threshold = threshold, type = "cen")
    d <- (cen - fit$centroid.overall)[aa, ]/fit$sd[aa]
    if(is.matrix(d)){o<-apply(abs(d), 1, max)}else{o<-max(abs(d))}
    oo <- order(-o)
    aa <- aa[oo]
    return(aa)
}
```

R Code B.8: Source Code: Preconditioning for Classification

```
Precond.p<-function(X.tr, X.val, y.tr, y.val, model="KNN",model2="PenLogReg",
    mod1.parms, mod2.parms){
    X.train<-X.tr
    y.train<-y.tr
    X.validation<-X.val
    y.validation<-y.val

    if(model=="SPC"){
        SPCA.out<-SPCA(X.mat=X.train, y.vec=y.train, gamma=mod1.parms, scale=T
            )
        if(!is.matrix(SPCA.out[[1]]) ){ #=="Gamma too large"
            return(list(Error=NA, Pred=NA, Dim.red2=NA, Fitted.Obj=NA,
                Selected.vars=NA))
        }
        SPC<-SPCA.out$UD.princomp
        spc.cor<-abs(apply(SPC,2, function(pc)cor(pc, y.train)))
        use.SPC<-which(spc.cor==max(spc.cor))
        log.reg.obj<-suppressWarnings(glm(y~x, data=data.frame(x=SPC[, use.SPC
            ], y=y.train), family = "binomial"))
        p.hat<-predict(log.reg.obj, type="response")
        ret<-SPCA.out$retained

    }else if(model=="KNN"){
        require(class)
        p.hat.win<-attributes(knn(train = X.train, test = X.train,cl = y.train
            ,
                    prob=T, k=mod1.parms))$prob
        p.hat<-1-abs(y.train-p.hat.win) #prob to belong to class 0
    } else if (model=="LogReg"){
        require(glmnet)
        log.reg.obj<-glmnet(X.train,as.factor(y.train),family="binomial",
                            alpha=1, lambda = 0)
        p.hat<-predict(log.reg.obj, newx=X.train, type="response")
    } else if (model=="LDA"){
        require(MASS)
        LDA.obj<-lda(as.factor(y.train)~X.train)
        p.hat<-predict(LDA.obj, newx=X.train)$posterior[,2] #class1
    } else if (model=="SVM"){
        require(e1071)
        SVM.obj<-svm(x=X.train, y=as.factor(y.train), type="C", kernel="radial
            ",
                    cost=mod1.parms[1], gamma=mod1.parms[2], probability=T)
        class.prob<-attributes(predict(SVM.obj, X.train, probability=T))$
            probabilities
        p.hat<-class.prob[,2] #class1
    }
    else if (model=="NSC"){
        require(pamr)
        capture.output(NSC.obj<-pamr.train(data=list(x=t(X.train), y=as.factor
            (y.train))))
        capture.output(class.prob<-pamr.predict(fit=NSC.obj, newx=t(X.train),
            threshold=mod1.parms, type = "posterior",
```

Stellenbosch University https://scholar.sun.ac.za


```
44                                  prior = NSC.obj$prior, threshold.scale = NSC.obj$
                                        threshold.scale))
45          p.hat<-class.prob[,2] #class1
46      }
47      #else if (model=="Boost"){
48      #     require(gbm)
49      #     boost.obj<-gbm(x=X.train, y=as.factor(y.train), distribution="
            bernoulli",
50      #                    n.trees=mod.parms[1],interaction.depth=mod.parms[2])
            #eg.5000, 4
51      #     p.hat<-predict(boost.obj,newdata=X.train,n.trees=mod.parms[1], type
            ="response")
52      #
53      # } else if (model=="RF"){
54      #     require(randomForest)
55      #     RF.obj<-randomForest(x=X.train, y=as.factor(y.train), ntree=mod.
            parms) #eg.2000
56      #     p.hat<-predict(RF.obj,newdata=X.train,n.tree=mod.parms, type="
            response")
57      #
58      # } else if (model=="NN"){
59      #     require(neuralnet)
60      #     NN.obj<-neuralnet(as.factor(y.train)~X.train, hidden=mod.parms,
            linear.output=FALSE) #eg. c(10,10,10)
61      #     p.hat<-compute(NN.obj, X.train)
62      # }
63      require(glmnet)
64      p.hat<-ifelse(p.hat==0, p.hat+0.00001, ifelse(p.hat==1, p.hat-0.00001, p.
            hat))
65      if (model2=="PenLogReg"){
66          y.hat<-log(p.hat/(1-p.hat))
67          if(var(y.hat)==0){print(list(p.hat, y.hat, model, model2, y.train))}
68          fitted.obj<-glmnet(X.train,y.hat,family="gaussian", alpha=1, lambda =
                mod2.parms, standardize = F)
69          vars.incl<-which(fitted.obj$beta != 0)
70          y.pred<-predict(fitted.obj, newx=X.validation)
71          p.pred<-exp(y.pred)/(1+exp(y.pred))
72          y.class<-ifelse(p.pred<0.5, 0, 1)
73      } else if (model2=="NSC"){
74          y.hat<-ifelse(p.hat<=0.5, 0, 1)
75          if(var(y.hat)==0){print(list(p.hat, y.hat, model, model2, y.train))}
76          require(pamr)
77          capture.output(fitted.obj<-pamr.train(data=list(x=t(X.train), y=y.hat)
                ))
78          capture.output(p.pred<-pamr.predict(fit=fitted.obj, newx=t(X.
                validation), threshold=mod2.parms, type = "posterior",
79                                  prior = fitted.obj$prior, threshold.scale = fitted
                                        .obj$threshold.scale))
80          vars.incl<-NSC.Incl.Vars(fitted.obj, t(X.train), mod2.parms)
81
82          y.class<-ifelse(p.pred[,2]<0.5, 0, 1)
83      }
84
85      mis.rate<- sum(y.class != y.validation)/length(y.validation)
86
87
88      return(list(Error=mis.rate, Pred=y.class,  Dim.red2=length(vars.incl),
89                  Fitted.Obj=fitted.obj, Selected.vars=vars.incl))
90
91  }
92
93  # Precond.p(matrix(rnorm(20), nr=5), matrix(rnorm(20), nr=5), sample(0:1, 5,
        replace = T),
94  #           sample(0:1, 5, replace=T), model="SPC", model2="NSC", 0.7, 1)
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.9: Source Code: Preconditioning Cross-Validation

```
1
2  cv.precond.p<-function(X.trval, y.trval,X.te, y.te, mod1.parm.range, mod2.parm
       .range, nfold=5, mod="SPC", mod2="PenLogReg"){
3      if(nfold<2){return("nfolds must be greater than 1")}
4      X.train_val<-X.trval
5      y.train_val<-y.trval
6      X.test<-X.te
7      y.test<-y.te
8      if (is.null(mod1.parm.range)){parms<-mod2.parm.range
9      }else if(is.list(mod1.parm.range)){parms<-expand.grid(mod1.parm.range
           [[1]], mod1.parm.range[[2]], mod2.parm.range)
10     }else {parms<-expand.grid(mod1.parm.range, mod2.parm.range)}
11     trails<-ifelse(is.matrix(parms),nrow(parms), length(parms))
12
13     # Initialise variables
14     error<-matrix(0, nc=nfold, nr=trails)
15     dim.red.spca<-matrix(0, nc=nfold, nr=trails)
16     dim.red.lasso<-matrix(0, nc=nfold, nr=trails)
17
18     options(warn=-1)
19     folds<-matrix(rep(1:nfold), nc=1, nr=nrow(X.train_val))
20     options(warn=0)
21
22     start<-Sys.time()
23     for (k in 1:nfold){
24         # Split non-test part into training and validation
25         X.train<-X.train_val[folds!=k,]
26         zero.var<-which(apply(X.train,2,var)<0.001)
27         X.train<-X.train[,-zero.var]
28         X.validation<-X.train_val[folds==k,]
29         X.validation<-X.validation[,-zero.var]
30         y.train<-y.train_val[folds!=k]
31         y.validation<-y.train_val[folds==k]
32         for(i in 1:trails){
33             if(is.vector(parms)){
34                 eval.precond<-Precond.p(X.tr=X.train, X.val=X.validation, y.tr
                       =y.train, y.val=y.validation,
35                                         model=mod,model2=mod2, mod1.parms=NULL
                                               , mod2.parms=parms[i])
36             }else{
37                 eval.precond<-Precond.p(X.tr=X.train, X.val=X.validation, y.tr
                       =y.train, y.val=y.validation,
38                                         model=mod,model2=mod2, mod1.parms=
                                               parms[i,-ncol(parms)],
39                                         mod2.parms=parms[i,ncol(parms)])
40             }
41             error[i, k]<-eval.precond$Error
42             dim.red.lasso[i,k]<-eval.precond$Dim.red2
43         }
44     }
45     end<-Sys.time()
46     runtime<-end-start
47     aves<-apply(error,1,mean, na.rm=T)
48     if(is.vector(parms)){
49         selected.parms<-na.omit(parms[min(which(aves==min(aves, na.rm = T)))])
50         names(selected.parms)<-c("Model 2 Parm")
51
52         ave.dim.red.lasso<-apply(dim.red.lasso,1,mean, na.rm=T)
53         ave.dim.red.select<-ave.dim.red.lasso[aves==min(aves, na.rm=T)]
54
55         # Fit Final Model
56         if(any(is.na(selected.parms))){return(paste("Invalid selected parms.
               See error rates:", aves))}
57         Final.Model.1<-Precond.p(X.tr=X.train, X.val=X.validation, y.tr=y.
```

Stellenbosch University https://scholar.sun.ac.za


```
            train , y.val=y.validation ,
58                                  model=mod, model2=mod2, mod1.parms=NULL,
59                                  mod2.parms=selected.parms)
60
61          X.test<-X.test[,-zero.var]
62          if(mod2=="NSC"){
63              capture.output(test.y.class<-pamr.predict(Final.Model.1$Fitted.Obj
                    , newx=t(X.test),threshold=selected.parms))
64          }else{
65              test.y.pred.1<-predict(Final.Model.1$Fitted.Obj, newx=X.test)
66              test.p.pred.1<-exp(test.y.pred.1)/(1+exp(test.y.pred.1))
67              test.y.class<-ifelse(test.p.pred.1<0.5, 0, 1)
68          }
69      }else {
70          selected.parms<-na.omit(parms[min(which(aves==min(aves, na.rm = T)))
                ,])
71          if(ncol(parms)>2){colnames(selected.parms)<-c("Model 1 Parm 1", "Model
                 1 Parm 2", "Model 2 Parm")
72          }else {colnames(selected.parms)<-c("Model 1 Parm", "Model 2 Parm")}
73
74          ave.dim.red.lasso<-apply(dim.red.lasso,1,mean, na.rm=T)
75          ave.dim.red.select<-ave.dim.red.lasso[aves==min(aves, na.rm=T)]
76
77          # Fit Final Model
78          if(any(is.na(selected.parms))){return(paste("Invalid selected parms.
                See error rates:", aves))}
79          Final.Model.1<-Precond.p(X.tr=X.train, X.val=X.validation, y.tr=y.
                train , y.val=y.validation ,
80                                  model=mod, model2=mod2, mod1.parms=selected.
                                        parms[1,-ncol(selected.parms)],
81                                  mod2.parms=selected.parms[1,ncol(selected.
                                        parms)])
82          X.test<-X.test[,-zero.var]
83          if(mod2=="NSC"){
84              capture.output(test.y.class<-pamr.predict(Final.Model.1$Fitted.Obj
                    , newx=t(X.test),threshold=selected.parms[1,ncol(selected.
                    parms)]))
85          }else{
86              test.y.pred.1<-predict(Final.Model.1$Fitted.Obj, newx=X.test)
87              test.p.pred.1<-exp(test.y.pred.1)/(1+exp(test.y.pred.1))
88              test.y.class<-ifelse(test.p.pred.1<0.5, 0, 1)
89          }
90      }
91
92
93
94      test.mis.rate<- sum(test.y.class != y.test)/length(y.test)
95
96
97      return(list(Runtime=runtime, Error.all=unique(aves), Optimal.Parms=
            selected.parms,
98                  Num.Final.Parms=ave.dim.red.select[1], Var.Select=Final.Model
                        .1$Selected.vars,
99                  Error.rate=test.mis.rate, Test.pred.class=test.y.class, Final.
                        Model=Final.Model.1))
100 }
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.10: Source Code: Compare All Preconditioning Models

```
Precond.Compare<-function(dat.set=data.set, Models=c("SPC","KNN", "LogReg", "LDA", "NSC"),
                          Model2="PenLogReg", Good.vars=1:50, verbose=F, folds=k){

    dat.X<-dat.set$X.mat
    dat.y<-dat.set$y.vec

    # Preprocess and partition
    Prep.dat<-prepro(X.mat=dat.X, y.vec=as.integer(dat.y))
    X.train_val<-Prep.dat$X.train
    X.test<-Prep.dat$X.validation
    y.train_val<-Prep.dat$y.train
    y.test<-Prep.dat$y.validation

    # Select a range of parameters to test in CV
    cors.sum<-summary(abs(apply(X.train_val, 2, function(x) cor(x, y.train_val))))
    spc.gamma<-as.vector(cors.sum) #seq(0.4, 0.6, length.out = 5)
    lasso.lambs<-seq(0.7, 0.07, length.out = 10) #seq(0.2, 0.07, length.out = 5)
    knn.neighb<-1:10
    svm.parms<-list(seq(0.1, 0.9, length.out = 10), seq(1, 101, length.out = 10))
    nsc.parms<-seq(0.1, 2.2, length.out = 10)
    if(Model2=="NSC"){mod2.parm<-nsc.parms}else{mod2.parm<-lasso.lambs}

    error<-vector(length = length(Models))
    vars.selected<-list()
    i<-0
    for (m in Models){
        i<-i+1
        if(verbose){print(m)}
        precond<-switch(m,
                "SPC"= cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                    mod="SPC", mod2=Model2, mod1.parm.range=spc.gamma, mod2.parm.range=mod2.parm,
                                    nfold=folds),
                "KNN"= cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                    mod="KNN", mod2=Model2, mod1.parm.range=knn.neighb, mod2.parm.range=mod2.parm,
                                    nfold=folds),
                "LogReg"=cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                      mod="LogReg", mod2=Model2, mod1.parm.range=NULL, mod2.parm.range=mod2.parm,
                                      nfold=folds),
                "LDA"=cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                   mod="LDA", mod2=Model2, mod1.parm.range=NULL, mod2.parm.range=mod2.parm,
                                   nfold=folds),
                "SVM"=cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                   mod="SVM", mod2=Model2, mod1.parm.range=svm.parms, mod2.parm.range=mod2.parm,
                                   nfold=folds),
                "NSC"=cv.precond.p(X.trval=X.train_val, y.trval=y.train_val, X.te=X.test, y.te=y.test,
                                   mod="NSC", mod2=Model2, mod1.parm.range=nsc.parms, mod2.parm.range=mod2.parm,
```

Stellenbosch University https://scholar.sun.ac.za


```
                                    nfold=folds)
        )
        # Save errors and selected vars
        error[i]<-precond$Error.rate
        vars.selected[[m]]<-precond$Var.Select

    }
    correct<-lapply(vars.selected, function(m)match(m, Good.vars))
    proportion.good<-lapply(correct, function(m) length(na.omit(m))/length(m))
    return(list(Final.Errors=error, Proportion.Good.Vars=proportion.good,
        Selected.Variables=vars.selected))
}
```

R Code B.11: Source Code: Compare Non-Preconditioning to Preconditioning

```
All.Models<- function(dat.set=Gen.data(250,1000,50, 0,0.1,1,0.1), r=50, k=5,
    verbose=F, p=2){
    dat.X<-dat.set$X.mat
    dat.y<-dat.set$y.vec

    # Preprocess and partition
    Prep.dat<-prepro(X.mat=dat.X, y.vec=as.integer(dat.y))
    X.train_val<-Prep.dat$X.train
    X.test<-Prep.dat$X.validation
    y.train_val<-Prep.dat$y.train
    y.test<-Prep.dat$y.validation

    zero.var<-which(apply(X.train_val,2,var)<0.001)
    X.train_val<-X.train_val[,-zero.var]
    X.test<-X.test[,-zero.var]

    if(p==1 ||is.null(p)){
        ############### Penalised Log Regression ##################################
        if(verbose){print("PLR")}
        library(glmnet)
        lam<-cv.glmnet(X.train_val, y.train_val, family = "binomial", alpha =
            1, nfolds=k)$lambda.min
        fitted.Log.Reg<-glmnet(X.train_val, y.train_val, family = "binomial",
            alpha = 1, lambda = lam)
        fitted.class<-predict(fitted.Log.Reg, newx = X.train_val, type="class"
            )
        coef<-fitted.Log.Reg$beta
        vars.incl.PLR<-which(coef!=0)
        num.vars<-length(vars.incl.PLR)
        prop.good.PLR<-length(na.omit(match(1:r, vars.incl.PLR)))/num.vars
        pred.class<-predict(fitted.Log.Reg, newx = X.test, type="class")
        test.err.PLR<-sum(pred.class != y.test)/length(y.test)

        ############### SPC ########################################################
        if(verbose){print("SPC")}
        SPCA.out<-cv.SPCA(X.mat=X.train_val, y.vec=y.train_val, k=k, scale = T
            )
        SPC<-SPCA.out$UD.princomp
        ret<-SPCA.out$retained
        spc.cor<-abs(apply(SPC,2, function(pc)cor(pc, y.train_val)))
        use.SPC<-which(spc.cor==max(spc.cor))
        require(glmnet)
        fitted.SPC<-suppressWarnings(glm(y~x, data=data.frame(x=SPC[, use.SPC
            ], y=y.train_val), family = "binomial"))
        X.test.trans<-X.test[,ret]%*%SPCA.out$V.rotation
        if(is.matrix(X.test.trans)){X.new<-X.test.trans[,use.SPC]}else{X.new<-
            X.test.trans}
        p.hat<-predict(fitted.SPC,newdata = data.frame(x=X.new), type="
            response")
```

Stellenbosch University https://scholar.sun.ac.za


```
42          p.hat<-ifelse(p.hat==0, p.hat+0.00001, ifelse(p.hat==1, p.hat-0.00001,
                p.hat))
43          y.pred<-ifelse(p.hat<0.5, 0, 1)
44          test.err.SPC<-sum(y.pred != y.test)/length(y.test)
45          vars.incl.SPC<-which(ret==T)
46          Correct.vars<-na.omit(match(vars.incl.SPC, 1:r ))
47          prop.good.SPC<-length(Correct.vars)/length(vars.incl.SPC)
48
49
50          ################ NSC ##################################################
51          if(verbose){print("NSC")}
52          require(pamr)
53          capture.output(NSC.obj<-pamr.train(data=list(x=t(X.train_val), y=as.
                factor(y.train_val))))
54          capture.output(NSC.obj.cv<-pamr.cv(NSC.obj, data=list(x=t(X.train_val)
                , y=as.factor(y.train_val)), nfold=5))
55          thresh<-NSC.obj.cv$threshold[max(which(NSC.obj.cv$error==min(NSC.obj.
                cv$error)))]
56          capture.output(fitted.NSC<-pamr.train(data=list(x=t(X.train_val), y=as
                .factor(y.train_val)), threshold = thresh))
57          train.err.NSC<-fitted.NSC$errors/length(y.train_val)
58          num.vars.nsc<-fitted.NSC$nonzero
59
60          vars.incl.nsc<-NSC.Incl.Vars(fitted.NSC, t(X.train_val), thresh)
61          prop.good.nsc<-length(na.omit(match(1:r, vars.incl.nsc)))/num.vars.nsc
62
63          capture.output(p.hat<-pamr.predict(fit=fitted.NSC, newx=t(X.test),
                type = "posterior",threshold= thresh ,
64                                  prior = fitted.NSC$prior, threshold.scale = fitted
                                      .NSC$threshold.scale)[,2])
65
66          pred.class.nsc<-ifelse(p.hat<0.5, 0, 1)
67          test.err.nsc<-sum(pred.class.nsc != y.test)/length(y.test)
68
69          ################ Combine results ##########################################
70          Test.Errors<-c(test.err.PLR, test.err.SPC, test.err.nsc)
71
72          No.Vars.Incl<-c(length(vars.incl.PLR), length(vars.incl.SPC), length(
                vars.incl.nsc))
73
74          Prop.Correct<-c(prop.good.PLR, prop.good.SPC, prop.good.nsc)
75          names(Test.Errors)<- names(No.Vars.Incl)<-  names(Prop.Correct)<- c("
                PLR", "SPC", "NSC" )
76
77      }
78      if(p==2 ||is.null(p)){
79          ################ Preconditioning (various techniques) ###################
80          if(verbose){print("Precond - PLR")}
81          Model1.results<-Precond.Compare(dat.set, Good.vars=1:r, Model2="
                PenLogReg", verbose = verbose, folds=k)
82          if(verbose){print("Precond - NSC")}
83          Model2.results<-Precond.Compare(dat.set, Good.vars=1:r, Model2="NSC",
                verbose = verbose, folds=k)
84
85          test.err.Precond.PLR<-Model1.results$Final.Errors
86          prop.good.Precond.PLR<-Model1.results$Proportion.Good.Vars
87          vars.incl.Precond.PLR<-Model1.results$Selected.Variables
88
89          test.err.Precond.NSC<-Model2.results$Final.Errors
90          prop.good.Precond.NSC<-Model2.results$Proportion.Good.Vars
91          vars.incl.Precond.NSC<-Model2.results$Selected.Variables
92
93          ################ Combine results ##########################################
94          if(is.null(p)){
95              Test.Errors<-c(Test.Errors, unlist(test.err.Precond.PLR), unlist(
                    test.err.Precond.NSC)
```

Stellenbosch University https://scholar.sun.ac.za


```
96              )
97
98              No.Vars.Incl<-c(No.Vars.Incl, unlist(lapply(vars.incl.Precond.PLR,
                    length)),
99                              unlist(lapply(vars.incl.Precond.NSC, length)))
100
101             Prop.Correct<-c(Prop.Correct, unlist(prop.good.Precond.PLR),
                    unlist(prop.good.Precond.NSC))
102             names(Test.Errors)<- names(No.Vars.Incl)<-  names(Prop.Correct)<-
103                 c("PLR", "SPC", "NSC" , paste(c("SPC","KNN","LogReg","LDA","
                        NSC"), "-PLR", sep=""),
104                   paste(c("SPC","KNN","LogReg","LDA","NSC"), "-NSC"))
105
106         } else {
107
108             Test.Errors<-c(unlist(test.err.Precond.PLR), unlist(test.err.
                    Precond.NSC)
109             )
110
111             No.Vars.Incl<-c(unlist(lapply(vars.incl.Precond.PLR, length)),
112                             unlist(lapply(vars.incl.Precond.NSC, length)))
113
114             Prop.Correct<-c(unlist(prop.good.Precond.PLR), unlist(prop.good.
                    Precond.NSC))
115             names(Test.Errors)<- names(No.Vars.Incl)<-  names(Prop.Correct)<-
116                 c(paste(c("SPC","KNN","LogReg","LDA","NSC"), "-PLR", sep=""),
117                   paste(c("SPC","KNN","LogReg","LDA","NSC"), "-NSC"))
118         }
119     }
120
121     return(data.frame(Test.Errors=Test.Errors, No.Vars.Incl=No.Vars.Incl, Prop
            .Correct=Prop.Correct))
122
123
124 }
125
126 # All.Models()
```

Stellenbosch University https://scholar.sun.ac.za

R Code B.12: Source Code: Running Simulation Study

```
# Simulated Data variables
n<-c(50,100,250,500,750,1000,2500)
p<-1000
r<-c(50,100,250,500)
mean1<-0
mean2<-c(0.1, 0.25, 0.5, 1)
var1<-1
var2<-c(0.1, 0.5, 1, 1.5, 2)

#Q1: How well does preconditioning compare to other tradiditional methods (all measures)?
Q1.parms<-expand.grid(250,1000, 50, 0,mean2,1,var2) #n,p,r,m1,m2,v1,v2

#Q2: How does it compare when sample size and SNR is altered?
Q2.parms<-expand.grid(n,1000, r, 0, 0.5,1,1) #n,p,r,m1,m2,v1,v2



Exp<-function(iterations, parameters, verbose=F, p=1){
    start<-Sys.time()
    no.sets<-nrow(parameters)
    nmod<-ifelse(is.null(p), 13, ifelse(p==1, 3, 10))
    ave.Test.Errors<-ave.No.Vars.Incl<-ave.Prop.Correct<-matrix(0, nrow=no.sets, ncol=nmod)
    var.Test.Errors<-var.No.Vars.Incl<-var.Prop.Correct<-matrix(0, nrow=no.sets, ncol=nmod)
    for(i in 1:no.sets){
        if(verbose){print(paste(parameters[i,]))}
        Test.Errors<-No.Vars.Incl<-Prop.Correct<-matrix(0, nrow=iterations, ncol=nmod)
        for(j in 1:iterations){
            print(paste(c('set', 'iter'), c(i,j)))
            set.seed(j)
            # Generate Data
            data.set<-Gen.data(n=parameters[i,1],p=parameters[i,2], r=parameters[i,3], m1=parameters[i,4],
                               m2=parameters[i,5], v1=parameters[i,6], v2=parameters[i,7])

            # Evaluate all models i.t.o :
            # - Error
            # - Num vars in final
            # - num correct in final
            results<-All.Models(dat.set=data.set, r=parameters[i,3], k=5, verbose=verbose, p=p)
            Test.Errors[j,]<-results$Test.Errors
            No.Vars.Incl[j,]<-results$No.Vars.Incl
            Prop.Correct[j,]<-results$Prop.Correct

        }

        #average of 3 measures over iterations
        ave.Test.Errors[i,]<-apply(Test.Errors , 2, mean)
        ave.No.Vars.Incl[i,]<-apply(No.Vars.Incl , 2, mean)
        ave.Prop.Correct[i,]<-apply(Prop.Correct , 2, mean)

        #variance of 3 measures over iterations
        var.Test.Errors[i,]<-apply(Test.Errors , 2, var)
        var.No.Vars.Incl[i,]<-apply(No.Vars.Incl , 2, var)
        var.Prop.Correct[i,]<-apply(Prop.Correct , 2, var)
    }
    colnames(ave.Test.Errors)<-colnames(ave.No.Vars.Incl)<-colnames(ave.Prop.Correct)<-rownames(results)
```

Stellenbosch University https://scholar.sun.ac.za


```
57      colnames(var.Test.Errors)<-colnames(var.No.Vars.Incl)<-colnames(var.Prop.
            Correct)<-rownames(results)
58
59      cat("\n")
60      end<-Sys.time()
61      return(list(Average.Err=round(ave.Test.Errors,4), Average.No.Vars=round(
            ave.No.Vars.Incl,4),
62                  Average.Prop.Correct=round(ave.Prop.Correct,4),
63                  Variance.Err=round(var.Test.Errors,4), Variance.No.Vars=round(
                        var.No.Vars.Incl,4),
64                  Variance.Prop.Correct=round(var.Prop.Correct, 4), Run.Time=end
                        -start))
65  }
66
67  library(lattice)
68
69  result1<-Exp(iterations = 10, parameters=Q1.parms, verbose=F, p=NULL)
70  result2<-Exp(iterations = 10, parameters=Q2.parms, verbose=F, p=NULL)
71  nan.pos<-which(is.nan(result2$Average.Prop.Correct))
72  if(!is.null(nan.pos)){
73      result2$Average.Prop.Correct[nan.pos]<-0
74      result2$Variance.Prop.Correct[nan.pos]<-0
75      }
76
77  rownames(result1$Average.Err)<-paste(Q1.parms[,5], Q1.parms[,7])
78
79  model.names<-rep(colnames(result1$Average.Err), each=nrow(Q1.parms))
80  model.names<-factor(model.names, levels=unique( as.character(model.names) ) )
81  cols<-c(rep(rainbow(3)[1],3), rep(rainbow(3)[2],5), rep(rainbow(3)[3],5))
82
83
84  ##############################################################################
85  #                       Study 1
86  ##############################################################################
87  ######################## 1. Averages ########################################
88  Q1.Ave.Err<-data.frame(Error=as.vector(result1$Average.Err),
89                         model=model.names,
90                         m2=Q1.parms[,5], v2=Q1.parms[,7])
91  # Q1.Ave.Err$Error <- reorder( Q1.Ave.Err$Error, Q1.Ave.Err$model )
92  Q1.No.Vars<-data.frame(No.Vars=as.vector(result1$Average.No.Vars),
93                         model=model.names,
94                         m2=Q1.parms[,5], v2=Q1.parms[,7])
95  result1$Average.Prop.Correct[which(is.nan(result1$Average.Prop.Correct))]<-0
96  Q1.Prop.Good<-data.frame(Prop.Good=as.vector(result1$Average.Prop.Correct),
97                         model=model.names,
98                         m2=Q1.parms[,5], v2=Q1.parms[,7])
99
100 barchart(model~Error|m2*v2, data=Q1.Ave.Err, #groups=model,
101           main="Error rates of models when distribution of Group 2 changes",
102           ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)
103 barchart(model~No.Vars|m2*v2, data=Q1.No.Vars,
104           main="No. Variables incl. in models when distribution of Group 2
                changes",
105           ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)
106 barchart(model~Prop.Good|m2*v2, data=Q1.Prop.Good,
107           main="% Signal Vars incl. in models when distribution of Group 2
                changes",
108           ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)
109
110
111 ######################## 1. Variance ########################################
112 Q1.S.Err<-data.frame(Error=as.vector(result1$Variance.Err),
113                         model=model.names,
114                         m2=Q1.parms[,5], v2=Q1.parms[,7])
115 Q1.S.No.Vars<-data.frame(No.Vars=as.vector(result1$Variance.No.Vars),
116                         model=model.names,
```

Stellenbosch University https://scholar.sun.ac.za


```
                          m2=Q1.parms[,5], v2=Q1.parms[,7])
result1$Variance.Prop.Correct[which(is.na(result1$Variance.Prop.Correct))]<-0
Q1.S.Prop.Good<-data.frame(Prop.Good=as.vector(result1$Variance.Prop.Correct),
                           model=model.names,
                           m2=Q1.parms[,5], v2=Q1.parms[,7])

barchart(model~Error|m2*v2, data=Q1.S.Err, #groups=model,
         main="Variance of Error rates of models when distribution of Group 2 changes",
         ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)
barchart(model~No.Vars|m2*v2, data=Q1.S.No.Vars,
         main="Variance of No. Variables incl. in models when distribution of Group 2 changes",
         ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)
barchart(model~Prop.Good|m2*v2, data=Q1.S.Prop.Good,
         main="Variance of % Signal Vars incl. in models when distribution of Group 2 changes",
         ylab="Variance of Group 2", xlab="Mean of Group 2", col=cols)



##############################################################################
#                        Study 2
##############################################################################
###################### 2. Averages ###########################################

model.names2<-rep(colnames(result2$Average.Err), each=nrow(Q2.parms))
model.names2<-factor(model.names2, levels=unique( as.character(model.names2) ))
Q2.Ave.Err<-data.frame(Error=as.vector(result2$Average.Err),
                       model=model.names2,
                       N=Q2.parms[,1], r=Q2.parms[,3])
Q2.No.Vars<-data.frame(No.Vars=as.vector(result2$Average.No.Vars),
                       model=model.names2,
                       N=Q2.parms[,1], r=Q2.parms[,3])
result2$Average.Prop.Correct[which(is.nan(result2$Average.Prop.Correct))]<-0
Q2.Prop.Good<-data.frame(Prop.Good=as.vector(result2$Average.Prop.Correct),
                         model=model.names2,
                         N=Q2.parms[,1], r=Q2.parms[,3])

barchart(model~Error|N*r, data=Q2.Ave.Err, #groups=model,
         main="Error rates of models when N and r change",
         ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)
barchart(model~No.Vars|N*r, data=Q2.No.Vars,
         main="No. Variables incl. in models when N and r change",
         ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)
barchart(model~Prop.Good|N*r, data=Q2.Prop.Good,
         main="% Signal Vars incl. in models when N and r change",
         ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)


###################### 2. Variance
    ############################################################


model.names2<-rep(colnames(result2$Variance.Err), each=nrow(Q2.parms))
model.names2<-factor(model.names2, levels=unique( as.character(model.names2) ))
Q2.S.Err<-data.frame(Error=as.vector(result2$Variance.Err),
                     model=model.names2,
                     N=Q2.parms[,1], r=Q2.parms[,3])
Q2.S.No.Vars<-data.frame(No.Vars=as.vector(result2$Variance.No.Vars),
                     model=model.names2,
                     N=Q2.parms[,1], r=Q2.parms[,3])
result2$Varince.Prop.Correct[which(is.nan(result2$Variance.Prop.Correct))]<-0
Q2.S.Prop.Good<-data.frame(Prop.Good=as.vector(result2$Variance.Prop.Correct),
```

Stellenbosch University https://scholar.sun.ac.za


```
177                                 model=model.names2,
178                                 N=Q2.parms[,1], r=Q2.parms[,3])
179
180 barchart(model~Error|N*r, data=Q2.S.Err, #groups=model,
181           main="Variance of Error rates of models when n and N change",
182           ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)
183 barchart(model~No.Vars|N*r, data=Q2.S.No.Vars,
184           main="Variance of No. Variables incl. in models when N and r change",
185           ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)
186 barchart(model~Prop.Good|N*r, data=Q2.S.Prop.Good,
187           main="Variance of % Signal Vars incl. in models when N and r change",
188           ylab="No. of Signal Vars", xlab="No. of Observations", col=cols)
```

Stellenbosch University https://scholar.sun.ac.za

B.2 Chapter 4 Code: Real-World Examples

R Code B.13: Source Code: Breast Cancer Experiment

```
# http://www.biolab.si/supp/bi-cancer/projections/info/BCGSE349_350.html

breast.cancer <- read.table('/Users/Jani/Google Drive/Thesis/Programme/Breast_Cancer.tab', header = T, sep = "\t", fill = TRUE)
breast.cancer[1:10,1:10]
which(breast.cancer[1,]!="continuous")
breast.cancer[,c(12626,12627)]
X.breast.cancer<-as.matrix(breast.cancer[-c(1,2),-c(12626,12627)])
class(X.breast.cancer) <- "numeric"

zero.var<-which(apply(X.breast.cancer,2,var)<0.001)
X.breast.cancer.nzv<-X.breast.cancer[,-zero.var]
X.breast.cancer.scal<-scale(X.breast.cancer.nzv)
library(caret)
breast.cancer.na <- preProcess(X.breast.cancer.scal, "knnImpute")
X.breast.cancer.imp <- predict(breast.cancer.na, X.breast.cancer)

Y.breast.cancer<-ifelse(breast.cancer[-c(1,2),c(12626)]=="resistant",1,0)
l.breast.cancer<-list(X.mat=X.breast.cancer.imp, y.vec=Y.breast.cancer)

set.seed(10)
breast.cancer.results<-All.Models(dat.set=l.breast.cancer, r=1, k=5, verbose=F, p=NULL)
breast.cancer.results
ord<-order(breast.cancer.results[,1])
breast.ord<-breast.cancer.results[ord,]
barplot(breast.ord[,2], names.arg = rownames(breast.ord), las=2)
points(breast.ord[,1])

# 2 Dim representation of Data
breast.SPCA<-cv.SPCA(X.breast.cancer.imp, Y.breast.cancer, k=2)
SPC<-breast.SPCA$UD.princomp
spc.cor<-abs(apply(round(SPC,4),2, function(pc)cor(pc, Y.breast.cancer)))
max.SPC<-which(spc.cor==max(spc.cor, na.rm=T))
second.SPC<-which(spc.cor==max(spc.cor[-max.SPC], na.rm=T))
plot(breast.SPCA$UD.princomp[,c(max.SPC, second.SPC)], main="SPCA of Breast Cancer Data", col=Y.breast.cancer+2,
     yaxt="n", xaxt="n", xlab="Max. correlated SPC",ylab="Second most correlated SPC", asp=1, pch=16)

# Clasisfication Results
barplot(breast.ord[,2], names.arg = rownames(breast.ord), las=2, axes=F, ylim=c(0,13000),
        main="Breast Cancer Data Results")
axis(side=4, at = pretty(range(breast.ord[,2])))
mtext("Number of Variables", side=4, line=3)
par(new = TRUE)
plot(breast.ord[,1], xaxt="n", col=3,  xlab=NA, pch=16, ylab="Misclassification Rate", ylim=c(0,0.5))
legend("topleft", legend=c("Error", "No. variables"), pch=c(16, 0), col=c(3, "grey"), cex=0.7)
```

R Code B.14: Source Code: Prostate Cancer Experiment

```
# http://www.biolab.si/supp/bi-cancer/projections/info/prostateGSE2443.html

prostate.cancer <- read.table('/Users/Jani/Google Drive/Thesis/Programme/prostata.tab', header = T, sep = "\t", fill = TRUE)
dim(prostate.cancer)
prostate.cancer[1:10,1:10]
```

Stellenbosch University https://scholar.sun.ac.za


```
which(prostate.cancer[1,]!="continuous")

X.prostate.cancer<-as.matrix(prostate.cancer[-c(1,2),-c(1,2)])
class(X.prostate.cancer) <- "numeric"

zero.var<-which(apply(X.prostate.cancer,2,var)<0.001)
if(length(zero.var)!=0){X.prostate.cancer.nzv<-X.prostate.cancer[,-zero.var]
}else{X.prostate.cancer.nzv<-X.prostate.cancer}
X.prostate.cancer.scal<-scale(X.prostate.cancer.nzv)
library(caret)
prostate.cancer.na <- preProcess(X.prostate.cancer.scal, "knnImpute")
X.prostate.cancer.imp <- predict(prostate.cancer.na, X.prostate.cancer)

Y.prostate.cancer<-ifelse(prostate.cancer[-c(1,2),1]=="normal",1,0)
l.prostate.cancer<-list(X.mat=X.prostate.cancer.imp, y.vec=Y.prostate.cancer)

prostate.cancer.results<-All.Models(dat.set=l.prostate.cancer, r=1, k=5,
    verbose=F, p=NULL)
prostate.cancer.results

# 2 Dim representation of Data
prostate.SPCA<-cv.SPCA(X.prostate.cancer.imp, Y.prostate.cancer, k=5)
SPC<-prostate.SPCA$UD.princomp
spc.cor<-abs(apply(SPC,2, function(pc)cor(pc, Y.prostate.cancer)))
max.SPC<-which(spc.cor==max(abs(spc.cor)))
second.SPC<-which(spc.cor==max(abs(spc.cor[-max.SPC])))

plot(prostate.SPCA$UD.princomp[,c(max.SPC,second.SPC)], main="SPCA of Prostate
     Cancer Data", col=Y.prostate.cancer+2,
     yaxt="n", xaxt="n", xlab="Max. correlated SPC",ylab="Second most
         correlated SPC", asp=1, pch=16)

# Clasisfication Results
ord<-order(prostate.cancer.results[,1])
prost.ord<-prostate.cancer.results[ord,]

barplot(prost.ord[,2], names.arg = rownames(prost.ord), las=2, axes=F, ylim=c
    (0,13000),
        main="Prostate Cancer Data Results")
axis(side=4, at = pretty(range(prost.ord[,2])))
mtext("Number of Variables", side=4, line=3)
par(new = TRUE)
plot(prost.ord[,1], xaxt="n", col=3,  xlab=NA, pch=16, ylab="Misclassification
     Rate", ylim=c(0,0.5))
legend("topleft", legend=c("Error", "No. variables"), pch=c(16, 0), col=c(3,
    1))
```

R Code B.15: Source Code: Medulloblastoma Experiment

```
# http://www.biolab.si/supp/bi-cancer/projections/info/meduloblastomiGSE468.
    html

medul.cancer <- read.table('/Users/Jani/Google Drive/Thesis/Programme/
    medulloblastoma.tab', header = T, sep = "\t", fill = TRUE)
dim(medul.cancer)
medul.cancer[,c(1466,1467)]

which(medul.cancer[1,]!="c")

X.medul.cancer<-as.matrix(medul.cancer[-c(1,2),-c(1466,1467)])
class(X.medul.cancer) <- "numeric"

zero.var<-which(apply(X.medul.cancer,2,var)<0.001)
if(length(zero.var)!=0){X.medul.cancer.nzv<-X.medul.cancer[,-zero.var]
}else{X.medul.cancer.nzv<-X.medul.cancer}
```

Stellenbosch University https://scholar.sun.ac.za


```
15 X.medul.cancer.scal<-scale(X.medul.cancer.nzv)
16 library(caret)
17 medul.cancer.na <- preProcess(X.medul.cancer.scal, "knnImpute")
18 X.medul.cancer.imp <- predict(medul.cancer.na, X.medul.cancer)
19
20 Y.medul.cancer<-ifelse(medul.cancer[-c(1,2),1466]=="Met",1,0)
21 l.medul.cancer<-list(X.mat=X.medul.cancer.imp, y.vec=Y.medul.cancer)
22 X.medul.cancer.outlier<-X.medul.cancer.imp[-15,]
23 Y.medul.cancer.outlier<-Y.medul.cancer[-15]
24 l.medul.cancer.outlier<-list(X.mat=X.medul.cancer.outlier, y.vec=Y.medul.
       cancer.outlier)
25
26 medul.cancer.results2<-All.Models(dat.set=l.medul.cancer.outlier, r=1, k=5,
       verbose=F, p=NULL)
27 cbind(medul.cancer.results[,1],medul.cancer.results2[,1])
28
29 # 2 Dim representation of Data
30 medul.SPCA<-cv.SPCA(X.medul.cancer.imp, Y.medul.cancer, k=2)
31 SPC<-medul.SPCA$UD.princomp
32 spc.cor<-abs(apply(SPC,2, function(pc)cor(pc, Y.medul.cancer)))
33 max.SPC<-which(spc.cor==max(abs(spc.cor)))
34 second.SPC<-which(spc.cor==max(abs(spc.cor[-max.SPC])))
35
36 plot(medul.SPCA$UD.princomp[-15,c(max.SPC, second.SPC)], main="SPCA of
       Medulloblastoma Data", col=Y.breast.cancer+2,
37       yaxt="n", xaxt="n", xlab="Max. correlated SPC",ylab="Second most
             correlated SPC", asp=1, pch=16)
38
39 # Clasisfication Results
40 ord<-order(medul.cancer.results[,1])
41 medul.ord<-medul.cancer.results[ord,]
42
43 barplot(medul.ord[,2], names.arg = rownames(medul.ord), las=2, axes=F, ylim=c
       (0,1500),
44          main="Medulloblastoma Data Results")
45 axis(side=4, at = pretty(range(medul.ord[,2])))
46 mtext("Number of Variables", side=4, line=3)
47 par(new = TRUE)
48 plot(medul.ord[,1], xaxt="n", col=3, xlab=NA, pch=16, ylab="Misclassification
       Rate", ylim=c(0,0.6))
49 legend("right", legend=c("Error", "No. variables"), pch=c(16, 0), col=c(3, 1))
```

Stellenbosch University https://scholar.sun.ac.za

Bibliography

Bair, E., Hastie, T., Paul, D. and Tibshirani, R. (2006). Prediction by supervised principal components. *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 119–137.

Bair, E. and Tibshirani, R. (2004). Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biology*, vol. 2, pp. 511–522.

Bellman, R.E. (1961). *Adaptive control processes*. Princeton University Press.

Breast Cancer (no date). University of Ljubljana, Faculty of Computer and Information Science, Bioinformatics Laboratory. Accessed: 2018-08-26.
Available at: http://www.biolab.si/supp/bi-cancer/projections/info/BCGSE349_350.html

Chang, J.C., Wooten, E.C., Tsimelzon, A., Hilsenbeck, S.G., Gutierrez, M.C., Elledge, R., Mohsin, S., Osborne, C.K., Chamness, G.C., Allred, D.C. *et al.* (2003). Gene expression profiling for the prediction of therapeutic response to docetaxel in patients with breast cancer. *The Lancet*, vol. 362, no. 9381, pp. 362–369.

Demšar, J., Leban, G. and Zupan, B. (2007). Freeviz-an intelligent multivariate visualization approach to explorative analysis of biomedical data. *Journal of Biomedical Informatics*, vol. 40, no. 6, pp. 661–671.

Donoho, D.L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, vol. 1, p. 32.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911.

Forina, M. (1991). UCI machine learning repository wine dataset. University of California, Irvine, School of Information and Computer Sciences. Accessed: 2018-04-12.
Available at: http://archive.ics.uci.edu/ml/datasets/Wine.

Friedman, J.H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175.

Stellenbosch University https://scholar.sun.ac.za

Friedman, J.H. (1997). On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55–77.

Gamage, P.T. (2017). Identification of brain tumor using image processing techniques. Tech. Rep., University of Moratuwa.
Available at: https://www.researchgate.net/publication/319623148_Identification_of_Brain_Tumor_using_Image_Processing_Techniques.

Ganesh, J., Reynolds, K.E., Luckett, M. and Pomirleanu, N. (2010). Online shopper motivations, and e-store attributes: An examination of online patronage behavior and shopper typologies. *Journal of Retailing*, vol. 86, no. 1, pp. 106 – 115. ISSN 0022-4359.
Available at: http://www.sciencedirect.com/science/article/pii/S0022435910000059.

Ghosh, D. (2002). Singular value decomposition regression models for classification of tumors from microarray experiments. In: *Proceedings of the 2002 Pacific Symposium on Biocomputing*, pp. 11462–11467.

Hastie, T., Taylor, J., Tibshirani, R. and Walther, G. (2007). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, vol. 1, pp. 1–29.

Hastie, T., Tibshirani, R., Botstein, D. and Brown, P. (2001). Supervised harvesting of expression trees. *Genome Biology*, vol. 1, pp. 1–12.

Hastie, T., Tibshirani, R., Eisen, M., Alizadeh, A., Levy, R., Staudt, L., Botstein, D. and Brown, P. (2000). Identifying distinct sets of genes with similar expression patterns via 'gene shaving'. *Genome Biology*, vol. 1, pp. 1–21.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer. ISBN 9780387848846.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. New York, NY: Springer.

Jiang, H., Deng, Y., Chen, H.-S., Tao, L., Sha, Q., Chen, J., Tsai, C.-J. and Zhang, S. (2004). Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC Bioinformatics*, vol. 5, pp. 1–12.

Jolliffe, I.T. (1982). A note on the use of principal components in regression. *Applied Statistics*, pp. 300–303.

Loehlin, J. (2004). *Latent Variable Models: An Introduction to Factor, Path, and Structural Equation Analysis*. 4th edn. Taylor & Francis. ISBN 9781410609823.
Available at: https://books.google.co.za/books?id=nyOoZue4LoAC.

Stellenbosch University https://scholar.sun.ac.za

MacDonald, T.J., Brown, K.M., LaFleur, B., Peterson, K., Lawlor, C., Chen, Y., Packer, R.J., Cogen, P. and Stephan, D.A. (2001). Expression profiling of medulloblastoma: Pdgfra and the ras/mapk pathway as therapeutic targets for metastatic disease. *Nature Genetics*, vol. 29, no. 2, p. 143.

Medulloblastoma (no date). University of Ljubljana, Faculty of Computer and Information Science, Bioinformatics Laboratory. Accessed: 2018-09-02.
Available at: http://www.biolab.si/supp/bi-cancer/projections/info/meduloblastomiGSE468.html

Murphy, K.P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press. ISBN 0262018020, 9780262018029.

Obringer, R. and Nateghi, R. (2018). Predicting urban reservoir levels using statistical learning techniques. *Scientific Reports*, vol. 8, no. 5164.

Paul, D., Bair, E., Hastie, T. and Tibshirani, R. (2008). Preconditioning for feature selection and regression in high-dimensional problems. *The Annals of Statistics*, vol. 36, no. 4, pp. 1595–1618.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572.

Piironen, J. and Vehtari, A. (2018). Iterative supervised principal components. In: *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, vol. 84 of *PMLR*. Lanzarote, Spain.

Pinto, D. (2016). Fastknn [online]. GitHub. Accessed: 2019-01-11.
Available at: https://github.com/davpinto/fastknn

Prostate (no date). University of Ljubljana, Faculty of Computer and Information Science, Bioinformatics Laboratory. Accessed: 2018-08-28.
Available at: http://www.biolab.si/supp/bi-cancer/projections/info/prostateGSE2443.html

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
Available at: https://www.R-project.org/

Singh, D., Febbo, P.G., Ross, K., Jackson, D.G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A.A., D'Amico, A.V., Richie, J.P. *et al.* (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, vol. 1, no. 2, pp. 203–209.

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6567–6572.

Stellenbosch University https://scholar.sun.ac.za

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2003). Class prediction by nearest shrunken centroids, with applications to dna microarrays. *Statistical Science*, pp. 104–117.

Zumel, N. (2016). Principal components regression, pt. 2: Y-aware methods [online]. Win-Vector blog. Accessed: 2018-05-12.
Available at: https://www.win-vector.com/blog/2016/05/pcr_part2_yaware